INTERPOLATION PAR DES SPLINES QUADRATIQUES QUI PRÉSERVENT LA FORME DES DONNÉES

par

TARIK MOUMINA

mémoire présenté au Département de mathématiques et d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, mars 1999

Le _____ , le jury suivant a accepté ce mémoire dans sa version finale.
date

Président-rapporteur: M. Jean-Pierre Dussault _____
Département de mathématiques et d'informatique

Membre: M. Jacques Dubois _____
Département de mathématiques et d'informatique

Membre: M. François Dubeau _____
Département de mathématiques et d'informatique

À mes parents et à mes deux frères

pour leur amour et soutien inconditionnels

tout au long de mes études.

À ma tante Houria et à son mari

pour leur aide financière et morale

durant ces dernières années.

SOMMAIRE

McAllister et Roulier (1981) ainsi que Schumaker (1983) ont proposé des algorithmes d'interpolation par des splines quadratiques qui préservent la forme des données (c'est-à-dire la convexité et la monotonie des données).

L'algorithme de McAllister et Roulier est une approche géométrique, tandis que celui de Schumaker est une approche analytique qui préserve la convexité des données et fait appel, au besoin, à son algorithme interactif pour préserver la monotonie.

DeVore et Yan (1986) ont fait une analyse de la convergence de ces algorithmes, et ont présenté deux façons d'améliorer l'ordre de convergence.

De plus, Lahtinen (1990) a amélioré la méthode de Schumaker pour préserver la monotonie des données sans avoir recours à l'algorithme interactif.

Plus tard, Iqbal (1992) a montré que pour une technique particulière d'estimation des pentes, les deux méthodes précédentes sont identiques, et que dans ce cas, l'algorithme de Schumaker génère automatiquement des fonctions interpolantes qui préservent la forme des données.

Dans ce mémoire, on présente, analyse et compare les résultats des études déjà faites, visant à trouver la solution du problème d'interpolation qui répond aux exigences fixées.

REMERCIEMENTS

Je voudrais exprimer tout d'abord mes remerciements aux professeurs qui m'ont fait l'honneur de participer au jury de ce mémoire.

Je remercie spécialement le professeur François Dubeau, mon directeur de recherche qui m'a appuyé par ses conseils, son soutien scientifique et financier, sa grande disponibilité et la confiance qu'il m'a accordée durant toute la période de la recherche.

J'exprime ma gratitude au Gouvernement du Québec et à l'Université de Sherbrooke pour les bourses qu'ils m'ont attribuées.

J'aimerais adresser mes vifs remerciements à ma mère, mon père et à mes deux frères pour leur amour inconditionnel et leurs encouragements.

Un grand merci aussi à ma tante Houria et à son mari qui m'ont soutenu et aidé à mener à bien mon travail.

Finalement, mes remerciements vont à toute personne qui, de près ou de loin, a contribué à la réalisation de ce travail, et en particulier à Florence Chiltz et Stéphane Ragot.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

En conception assistée par ordinateur, en infographie et dans d'autres domaines scientifiques, le problème de construire une courbe lisse qui passe par des points donnés a été posé à plusieurs reprises, et une technique fréquemment utilisée est l'interpolation par une spline cubique. Cependant, les splines cubiques produisent parfois des oscillations indésirables qui rendent la courbe visuellement *non-plaisante.*

De même, souvent dans les expériences physiques, les données ont des formes particulières, telles que la monotonie et/ou la convexité, et on voudrait décrire une fonction interpolante qui préserve ces formes. Les algorithmes qui préservent la forme des données sont typiquement basés sur des algorithmes d'interpolation par des splines. Ces deux types d'algorithmes introduisent des nœuds additionnels et/ou modifient les pentes prescrites afin de préserver la forme requise.

De Boor ([5], 1978) a proposé un algorithme *τ-spline* qui préserve la convexité des données en insérant au plus un point additionnel entre chaque paire de points donnés, mais la monotonie n'est pas garantie. Fritsch et Carlson ([9], 1980) ont décrit un algorithme où la spline cubique, fonction d'interpolation initiale, est modifiée en changeant les valeurs des dérivées de cette fonction pour produire une spline cubique, fonction d'interpolation monotone. Costantini ([4], 1988) a développé la méthode de Fritsch et Carlson pour inclure des fonctions d'interpolation de degré arbitraire, et a proposé une nouvelle technique pour

l'estimation des pentes. Les méthodes de Beatson et Wolkowicz ([2], 1989) ainsi que celle de Yan ([19], 1987) préservent la monotonie des données et représentent une alternative pour la méthode de Fritsch et Carlson. Les valeurs des dérivées ne sont pas modifiées. En effet, quand les valeurs des dérivées n'assurent pas la monotonie sur un sous-intervalle, au lieu de changer les pentes, des nœuds additionnels sont insérés dans ce sous-intervalle.

McAllister et Roulier ([15][16], 1981) ont proposé un algorithme où, pour préserver la monotonie et/ou la convexité des données, le choix des pentes et des nœuds est basé sur des arguments géométriques. Le polynôme quadratique $\mathcal{C}^1$-par morceaux qui en résulte est construit à partir des polynômes de Bernstein [14]. Schumaker ([17], 1983) a présenté un algorithme, utilisant des polynômes quadratiques $\mathcal{C}^1$-par morceaux, qui préserve la convexité en ajoutant un seul nœud, quand c'est nécessaire, dans chaque sous-intervalle de données. Il fait appel au besoin à son algorithme interactif pour préserver la monotonie. Ce dernier algorithme permet à son utilisateur de modifier la fonction interpolante en changeant les pentes et/ou l'emplacement des nœuds. Lahtinen ([12], 1990) a amélioré la méthode de Schumaker pour préserver la monotonie des données sans avoir recours à l'algorithme interactif. Iqbal ([10], 1992) a présenté une comparaison de l'algorithme de McAllister et Roulier avec celui de Schumaker. Il a montré que pour un choix particulier des pentes, ces deux algorithmes sont similaires. De plus, quand les données sont convexes, Frey ([8], 1985) a amélioré les pentes de façon itérative afin de produire des courbes visiblement plus plaisantes. Pour les calculs d'erreurs, DeVore et Yan ([6], 1986) ont présenté une analyse de la convergence des splines quadratiques qui préservent la forme des données.

Dans ce mémoire, on s'intéresse en particulier aux splines quadratiques interpolantes qui préservent la forme des données. Toutes les méthodes et tous les algorithmes sont présentés dans une notation commune.

L'organisation du mémoire est la suivante. La première partie est un aperçu sur les études

et les algorithmes déjà faits; des remarques sont ensuite exposées. Dans le deuxième chapitre, on présente l'algorithme de McAllister et Roulier ; puis dans le troisième chapitre, l'algorithme de Schumaker. Ensuite les modifications faites par Lahtinen et Iqbal à cet algorithme sont présentées au quatrième chapitre. Enfin dans le cinquième chapitre, on présente une analyse de la convergence de ces méthodes.

CHAPITRE 1

ÉNONCÉ DU PROBLÈME

1.1 Introduction

Dans ce mémoire, on s'intéresse à la solution du problème d'interpolation suivant :
On se donne

- une subdivision $\{x_i\ ;\ 0 \leq i \leq n\}$ de l'intervalle $[a, b]$ de $\mathbb{R}$, telle que :
  $a = x_0 < \ldots < x_i < \ldots < x_n = b$ où $n \in \mathbb{N}$;
- une suite de points $\{(x_i, y_i)\}_{i=0}^{n}$ où les y_i sont des valeurs données ou proviennent d'une fonction $f : [a, b] \longrightarrow \mathbb{R}$ avec $y_i = f(x_i)$.

On cherche alors une fonction régulière p telle que

$$p(x_i) = y_i, \qquad i = 0, 1, \ldots, n. \tag{1.1}$$

En d'autres termes, il faut trouver une courbe lisse qui passe par tous les points (x_i, y_i), $i = 0, 1, \ldots, n$. Il y a plusieurs méthodes pour résoudre ce problème. Une des plus connues

est celle du polynôme d'interpolation de Lagrange. Cependant dans ce mémoire, on s'intéresse seulement aux méthodes qui préservent la forme des données, dans le sens où, dans les intervalles où les données sont monotones croissantes ou décroissantes, p doit avoir les mêmes propriétés. De façon similaire, dans les intervalles où les données sont convexes ou concaves, p doit l'être aussi.

1.2 Notations et définitions

On note par :

- I_i le i-ème intervalle de la subdivision

$$I_i = [x_{i-1}, x_i], \qquad i = 1, \ldots, n, \tag{1.2}$$

- $x_{i-\frac{1}{2}}$ le milieu de l'intervalle I_i

$$x_{i-\frac{1}{2}} = \frac{x_{i-1} + x_i}{2}, \qquad i = 1, \ldots, n, \tag{1.3}$$

- δ_i la pente de la corde reliant les points (x_{i-1}, y_{i-1}) et (x_i, y_i)

$$\delta_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \qquad i = 1, \ldots, n, \tag{1.4}$$

Définition 1.2.1 *Les données* $\{(x_i, y_i)\}_{i=0}^n$ *sont dites :*

- *croissantes si* $\delta_i \geq 0, i = 1, \ldots, n,$
- *convexes si* $\delta_i \geq \delta_{i-1}, i = 2, \ldots, n,$

et si les inégalités sont strictes on dit que les données sont strictement croissantes ou strictement convexes selon le cas.

Elles sont dites :

- *décroissantes si* $\delta_i \leq 0, i = 1, \ldots, n,$

- *concaves si* $\delta_i \leq \delta_{i-1}, i = 2, \ldots, n,$

et de même si les inégalités sont strictes on dit que les données sont strictement décroissantes ou strictement concaves selon le cas.

Exemple 1.2.2 *La Figure 1.1 présente l'exemple donné par* `Akima` *[1], où les* x_i *et* y_i *sont donnés dans le tableau suivant*

Tableau 1.1 – *L'exemple donné par* `Akima`

x_i	0	2	3	5	6	8	9	11	12	14	15
y_i	10	10	10	10	10	10	10.5	15	50	60	85

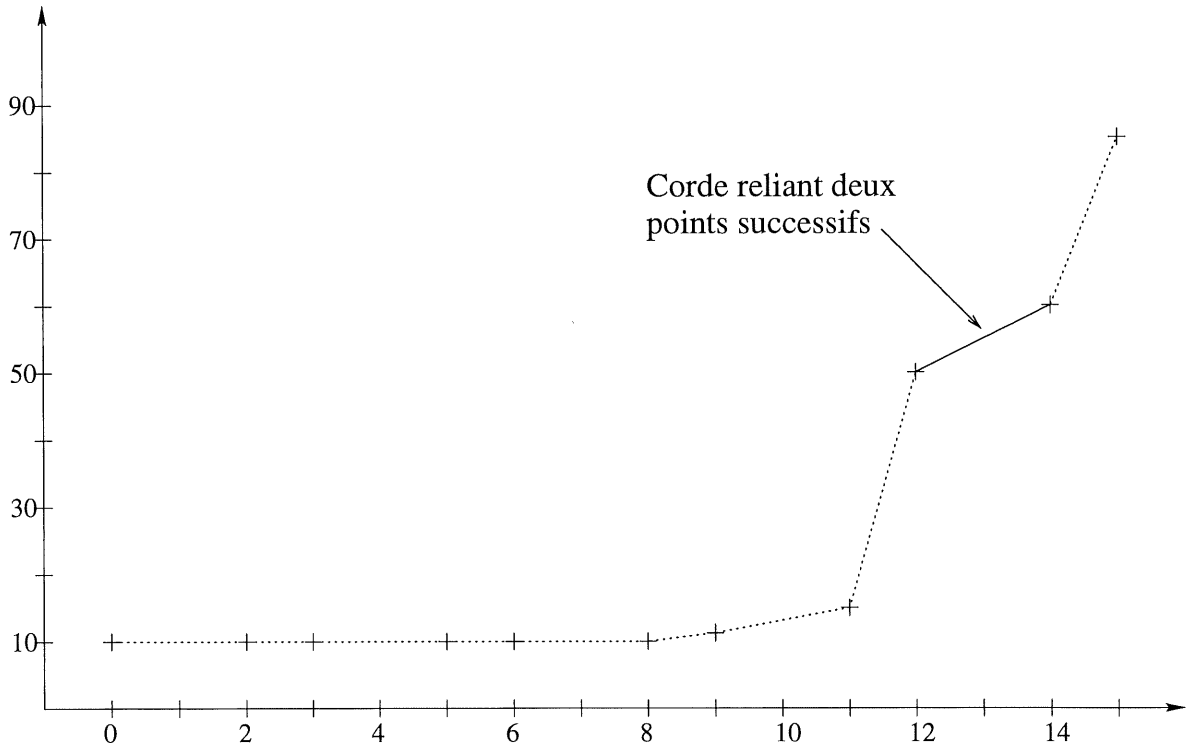


Figure 1.1 – *Exemple d'*`Akima`

Remarque 1.2.3

1. *Dans l'exemple d'*`Akima`*, les données sont croissantes. De plus elles sont strictement croissantes sur l'intervalle* $[8, 15]$. *Aussi, les données sont convexes sur l'intervalle* $[0, 12]$, *strictement convexes sur* $[12, 15]$ *et strictement concaves sur* $[11, 14]$.
2. *Le polynôme d'interpolation de Lagrange est un exemple de fonction qui interpole les données sans résoudre le problème, puisque ce polynôme ne respecte pas la forme des données (voir la Figure 1.2).*

Il existe différentes méthodes, basées sur l'interpolation par des splines, qui préservent la forme des données. À cette fin, ces méthodes introduisent des nœuds additionnels ou modifient les pentes prescrites. Dans ce mémoire, on s'intéresse en particulier aux méthodes qui utilisent des splines quadratiques.

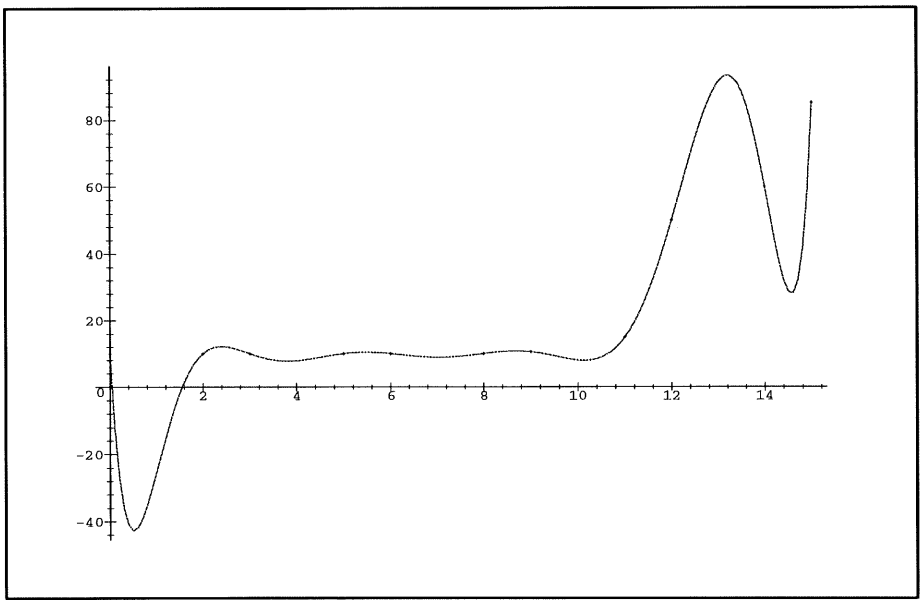


Figure 1.2 – *Polynôme d'interpolation de Lagrange*

1.3 Splines quadratiques

Soit $a = x_0 < x_1 < \ldots < x_n = b$ une partition de $[a, b]$. Une spline quadratique p est une fonction $p \in \mathcal{C}^1[a, b]$ telle que la restriction de p sur chaque sous-intervalle $[x_{i-1}, x_i]$ est un polynôme de degré ≤ 2 (voir [7][18]). Les points $\{x_i\}_{i=0}^n$ sont appelés **nœuds**.

1.4 Problème d'interpolation d'Hermite

Soient d_i les valeurs des dérivées aux points $(x_i, y_i), i = 0, 1, \ldots, n$. Le problème d'interpolation d'Hermite consiste à trouver une fonction $p \in \mathcal{C}^1[a, b]$ telle que :

$$p(x_i) = y_i, \qquad p'(x_i) = d_i, \qquad i = 0, 1, \ldots, n. \tag{1.5}$$

Définition 1.4.1 *Soit p une solution du problème d'interpolation d'Hermite. On dit que p préserve*

(i) la monotonie : si p est croissante sur les intervalles où les données sont croissantes, et décroissante sur les intervalles où les données sont décroissantes,

(ii) la convexité : si p' est croissante sur les intervalles où les données sont convexes, et décroissante sur les intervalles où les données sont concaves,

(iii) la forme des données : si p préserve à la fois la monotonie et la convexité.

Remarque 1.4.2

1. *Pour que la fonction d'interpolation p préserve la monotonie sur l'intervalle $[x_{i-1}, x_i]$, $p'(x)$ doit avoir le même signe que δ_i.*
2. *Souvent dans ce mémoire, on notera par I le i-ème intervalle $I_i = [x_{i-1}, x_i]$.*

1.5 Énoncé du Problème

Le problème consiste à trouver une spline quadratique, solution du problème d'interpolation d'Hermite (1.5), avec au plus un nœud supplémentaire ξ_i dans chaque sous-intervalle $]x_{i-1}, x_i[$, $i = 1, \ldots, n$.

La spline dépend du choix des pentes d_i, $i = 1, \ldots, n$ et des nœuds additionnels. Le but est de savoir comment choisir ces pentes et où placer les nœuds pour que la spline interpolante préserve la forme des données.

CHAPITRE 2

ALGORITHME DE McALLISTER ET ROULIER

En 1981, McAllister et Roulier [15][16] ont proposé un algorithme qui produit une fonction d'interpolation locale, une spline quadratique continuement différentiable qui préserve la forme des données, en insérant au plus deux nœuds entre chaque paire de données adjacentes. La sélection des pentes et des nœuds est basée sur les arguments géométriques décrits ci-après et la spline quadratique est construite en utilisant les polynômes de Bernstein [14].

2.1 Polynômes de Bernstein

Soient $S = (x_{i-1}, y_{i-1})$ et $T = (x_i, y_i)$ deux points donnés. Notons par I l'intervalle $I_i = [x_{i-1}, x_i]$ et soit $U = (u_1, u_2)$ un point arbitraire tel que :

$$u_1 = \frac{x_{i-1} + x_i}{2} = x_{i-\frac{1}{2}}. \tag{2.1}$$

Soit g la spline de degré un qui passe par les points S, U et T. Elle vérifie

$$g(x_{i-1}) = y_{i-1}, \qquad g(u_1) = u_2, \qquad g(x_i) = y_i, \tag{2.2}$$

et

$$g'(x) = \begin{cases} \dfrac{u_2 - y_{i-1}}{u_1 - x_{i-1}} = 2\,\dfrac{u_2 - y_{i-1}}{x_i - x_{i-1}} & \text{si} \quad x \in [x_{i-1}, u_1], \\ \dfrac{y_i - u_2}{x_i - u_1} = 2\,\dfrac{y_i - u_2}{x_i - x_{i-1}} & \text{si} \quad x \in [u_1, x_i], \end{cases} \tag{2.3}$$

en utilisant les dérivées à droite ou à gauche aux extrémités selon le cas.
Soit $B_2(g)$ le polynôme de Bernstein de degré deux associé à g sur l'intervalle I qu'on notera dorénavant $B_2(S, U, T)$ (voir la Figure 2.1).

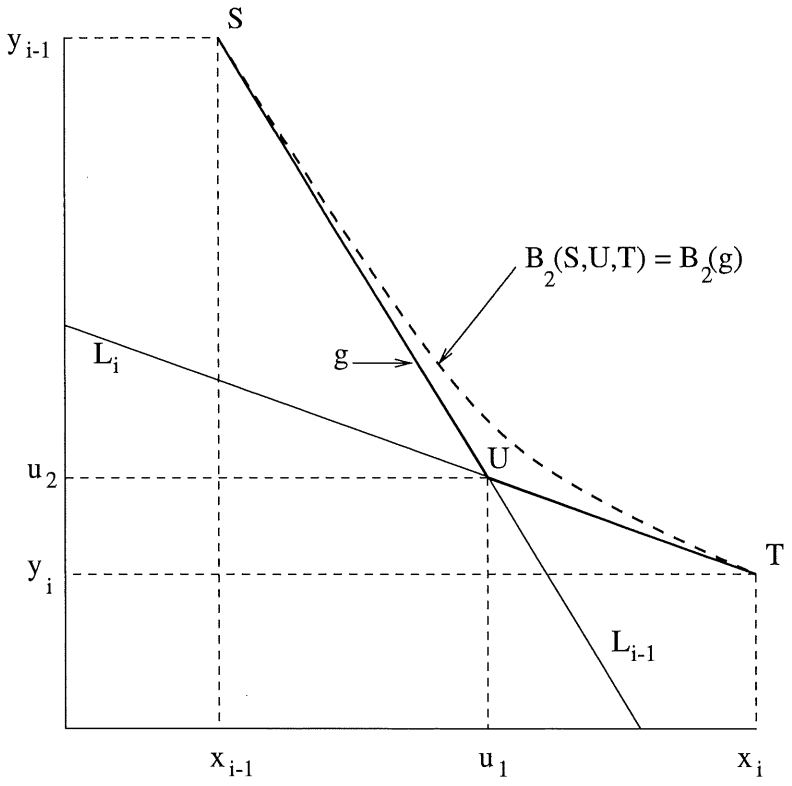


Figure 2.1 – *Polynômes de Bernstein*

$$
\begin{aligned}
B_2(S,U,T)(x) &= B_2(g)(x) \\
&= g(x_{i-1}) \left(\frac{x_i - x}{x_i - x_{i-1}}\right)^2 + 2g(u_1) \left(\frac{x_i - x}{x_i - x_{i-1}}\right)\left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right) \\
&\qquad + g(x_i) \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)^2 \\
&= \frac{1}{(x_i - x_{i-1})^2} [g(x_{i-1})(x_i - x)^2 + 2g(u_1)(x - x_{i-1})(x_i - x) \\
&\qquad + g(x_i)(x - x_{i-1})^2]. \qquad (2.4)
\end{aligned}
$$

Proposition 2.1.1 *Les polynômes de Bernstein ont les propriétés suivantes*

(1) $B_2(g)(x_{i-1}) = y_{i-1}$ *et* $B_2(g)(x_i) = y_i$,

(2) $B_2'(g)(x_{i-1}) = g'(x_{i-1}) = \dfrac{u_2 - y_{i-1}}{u_1 - x_{i-1}} = 2\,\dfrac{u_2 - y_{i-1}}{x_i - x_{i-1}}$ *et*

$B_2'(g)(x_i) = g'(x_i) = \dfrac{y_i - u_2}{x_i - u_1} = 2\,\dfrac{y_i - u_2}{x_i - x_{i-1}}$,

(3) si g *est monotone sur* I, *alors* $B_2(g)$ *est monotone sur* I,

(4) si g *est convexe (concave) sur* I, *alors* $B_2(g)$ *est convexe (concave) sur* I.

Démonstration La propriété (1) est immédiate à partir de (2.4) et (2.2). Pour montrer les propriétés (2) et (3), il suffit de remarquer que

$$
B_2'(S,U,T)(x) = \left(\frac{x_i - x}{x_i - x_{i-1}}\right) g'(x_{i-1}) + \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right) g'(x_i) \qquad (2.5)
$$

en utilisant (2.4) et (2.3). Finalement, comme

$$
B_2''(S,U,T)(x) = \frac{1}{x_i - x_{i-1}} (g'(x_i) - g'(x_{i-1})) \qquad (2.6)
$$

on en déduit la propriété (4). ■

Remarque 2.1.2 $B_2(g)$ *préserve la forme de* g *et* $B_2(g)$ *est continuement dérivable sur* I, *alors que* g *ne l'est pas. En utilisant cette propriété, on construit une spline* g *de degré un sur* I, *de pentes* d_{i-1} *en* S *et* d_i *en* T, *et telle que l'intervalle* I *puisse être partitionné en sous-intervalles de façon à ce que le polynôme quadratique par morceaux* p, *satisfaisant* $p = B_2(g)$ *sur chaque sous-intervalle, soit une spline quadratique, c'est-à-dire* $p \in \mathcal{C}^1[x_{i-1}, x_i]$.

2.2 Méthode de McAllister et Roulier

Soient $S = (x_{i-1}, y_{i-1})$ et $T = (x_i, y_i)$ deux points donnés décroissants ayant les pentes d_{i-1} et d_i respectivement. Soit $\mathcal{R}$ l'ensemble des points

$$\mathcal{R} = \{(x, y) : x_{i-1} \leq x \leq x_i \text{ et } y_i \leq y \leq y_{i-1}\} - \{S, T\}. \tag{2.7}$$

Notons par **M** le segment de droite d'extrémités

$$F = \left(x_{i-\frac{1}{2}}, y_{i-1}\right) \qquad \text{et} \qquad G = \left(x_{i-\frac{1}{2}}, y_i\right). \tag{2.8}$$

Soient L_{i-1} la droite passant par S de pente d_{i-1}, et L_i la droite passant par T de pente d_i. Si ces deux droites ont un point d'intersection, ce point sera noté $Z = (z_1, z_2)$. Notons par A le point de coordonnées (x_{i-1}, y_i) et par B le point de coordonnées (x_i, y_{i-1}), de façon à ce que l'ensemble $\mathcal{R}$ puisse être représenté par le rectangle $ASBT$ (voir la Figure 2.2). Pour les troisième et quatrième cas, on notera $C = (c_1, c_2)$, respectivement

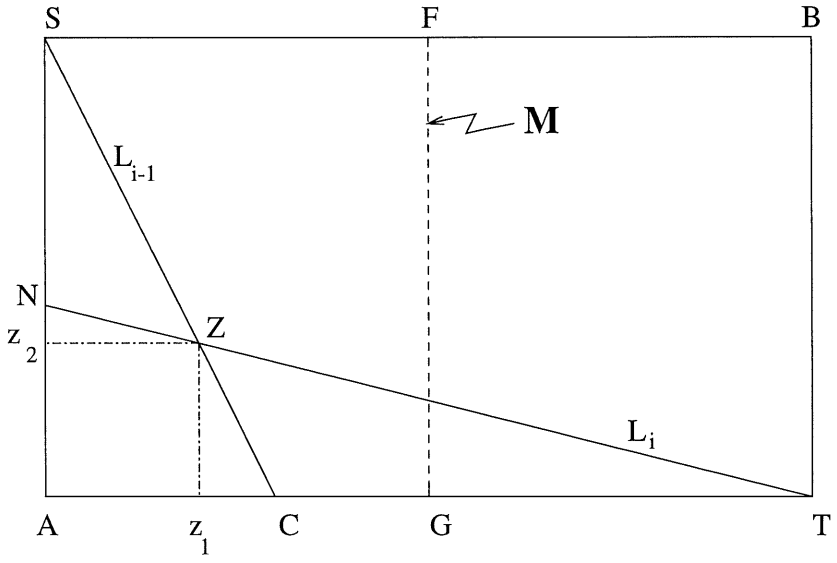


Figure 2.2 – *Méthode de McAllister et Roulier :* `Définitions`

$N = (n_1, n_2)$, le point d'intersection de la droite L_{i-1}, respectivement L_i, et la frontière de l'ensemble $\mathcal{R}$.

McAllister et Roulier ont distingué quatre cas qui peuvent survenir généralement. Cependant ils ont montré qu'on pouvait se limiter aux deux premiers cas pour un choix particulier des pentes.

2.2.1 Premier cas

Le point $Z = (z_1, z_2)$ d'intersection des droites L_{i-1} et L_i est tel que $z_1 \in]x_{i-1}, x_i[$ (voir la Figure 2.3).

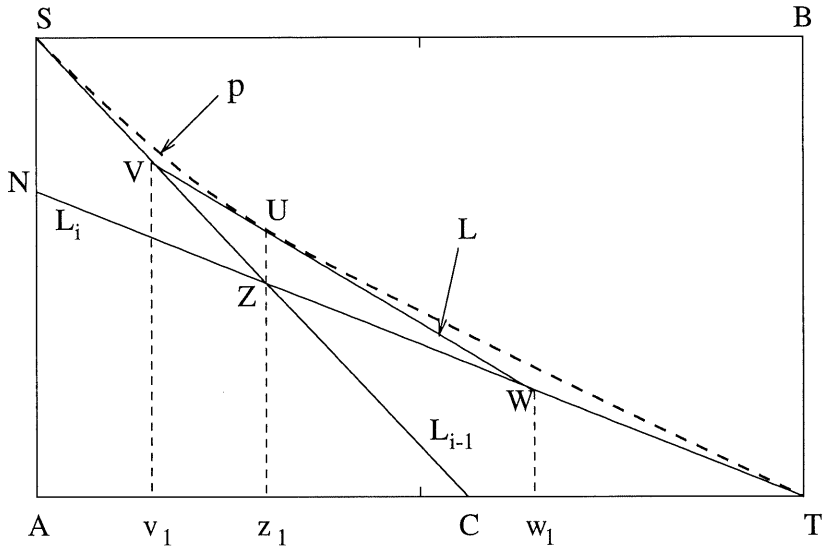


Figure 2.3 – *Méthode de McAllister et Roulier :* $1^{\texttt{er}}$ `cas`

L'algorithme insère un nœud additionnel en $x = \xi_i = z_1$, où

$$\xi_i = z_1 = x_i - \frac{\delta_i - d_{i-1}}{d_i - d_{i-1}}(x_i - x_{i-1}) = x_{i-1} + \frac{d_i - \delta_i}{d_i - d_{i-1}}(x_i - x_{i-1}) \tag{2.9}$$

Prenons

$$V = (v_1, v_2) = \left(\frac{x_{i-1} + \xi_i}{2}, L_{i-1}\left(\frac{x_{i-1} + \xi_i}{2}\right)\right) \tag{2.10}$$

et

$$W = (w_1, w_2) = \left(\frac{x_i + \xi_i}{2}, L_i \left(\frac{x_i + \xi_i}{2} \right) \right) \tag{2.11}$$

On définit $\eta_i = L(\xi_i)$, où L est la droite passant par V et W. La spline quadratique p est définie sur $[x_{i-1}, x_i]$ en utilisant le nœud supplémentaire $U = (\xi_i, \eta_i)$:

$$\begin{aligned} p(x) &= \begin{cases} B_2(S,V,U)(x), & x \in [x_{i-1}, \xi_i] \\ B_2(U,W,T)(x), & x \in [\xi_i, x_i] \end{cases} \\ &= \begin{cases} \dfrac{1}{(\xi_i - x_{i-1})^2}[y_{i-1}(\xi_i - x)^2 + 2v_2(x - x_{i-1}) \\ \qquad \times(\xi_i - x) + \eta_i(x - x_{i-1})^2], & x \in [x_{i-1}, \xi_i] \\ \dfrac{1}{(x_i - \xi_i)^2}[\eta_i(x_i - x)^2 + 2w_2(x - \xi_i) \\ \qquad \times(x_i - x) + y_i(x - \xi_i)^2], & x \in [\xi_i, x_i] \end{cases} \end{aligned} \tag{2.12}$$

Proposition 2.2.1 *La pente au nœud additionnel est égale à celle de la corde reliant les deux points S et T :* $p'(z_1) = \delta_i$.

Démonstration La pente au point z_1 est égale à celle de la droite L, or

$$L(x) = v_2 + \frac{w_2 - v_2}{w_1 - v_1}(x - v_1),$$

où $V = (v_1, v_2)$ et $W = (w_1, w_2)$ sont données par les équations (2.10) et (2.11) :

$$v_1 = \tfrac{1}{2}(x_{i-1} + \xi_i), \qquad v_2 = y_{i-1} + d_{i-1}(v_1 - x_{i-1}) = y_{i-1} + \tfrac{1}{2}d_{i-1}(\xi_i - x_{i-1}),$$

$$w_1 = \tfrac{1}{2}(\xi_i + x_i), \qquad w_2 = y_i + d_i(w_1 - x_i) = y_i + \tfrac{1}{2}d_i(\xi_i - x_i).$$

$$\begin{aligned} w_2 - v_2 &= y_i - y_{i-1} + \frac{1}{2}d_i(\xi_i - x_i) - \frac{1}{2}d_{i-1}(\xi_i - x_{i-1}) \\ &= y_i - y_{i-1} - \frac{1}{2}d_i(x_i - x_{i-1}) + \frac{1}{2}(d_i - d_{i-1})(\xi_i - x_{i-1}) \\ w_1 - v_1 &= \frac{1}{2}(x_i - x_{i-1}) \end{aligned}$$

La pente de la droite L est alors donnée par :

$$\frac{w_2 - v_2}{w_1 - v_1} = 2\delta_i - d_i + \frac{(d_i - d_{i-1})(\xi_i - x_{i-1})}{x_i - x_{i-1}},$$

et en remplaçant ξ_i par (2.9), on obtient le résultat. ■

2.2.2 Deuxième cas

Dans ce cas, on distingue trois situations différentes :

(1) $d_{i-1} \geq 0$ et $d_i \geq 0$,

(2) $d_{i-1}d_i < 0$ et le point Z est tel que $z_1 \notin]x_{i-1}, x_i[$,

(3) L_{i-1} et L_i n'ont pas d'intersection dans $\mathcal{R}$, mais par contre les deux droites coupent le segment $\mathbf{M}$ dans $\mathcal{R}$.

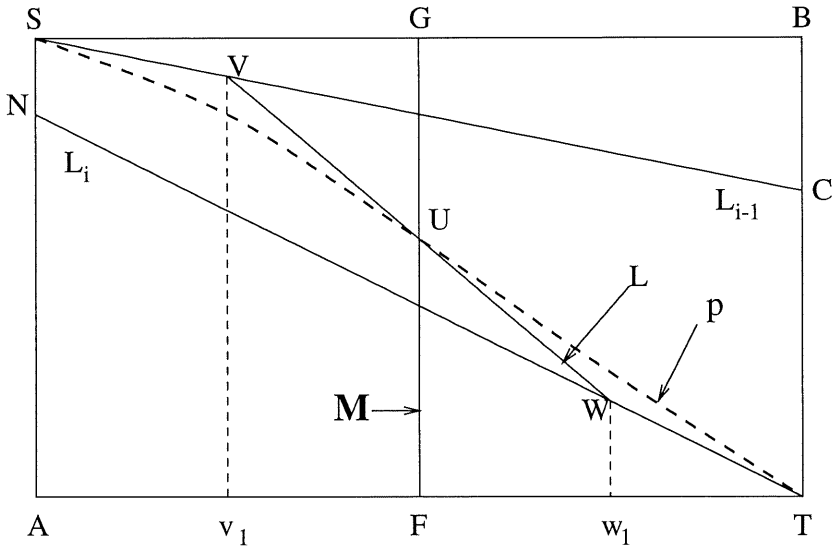


Figure 2.4 – *Méthode de McAllister et Roulier :* 2ème cas

Cette dernière situation est représentée par la Figure 2.4. On introduit un seul nœud additionnel

$$\xi_i = x_{i-\frac{1}{2}} \tag{2.13}$$

et on définit les points V, W, U et la spline quadratique p comme dans le cas précédent.

2.2.3 Troisième cas

C'est le cas où une seule des droites L_{i-1} et L_i intersecte le segment $\mathbf{M}$ dans $\mathcal{R}$. Sans perte de généralité, on suppose que c'est la droite L_{i-1} qui coupe le segment $\mathbf{M}$.

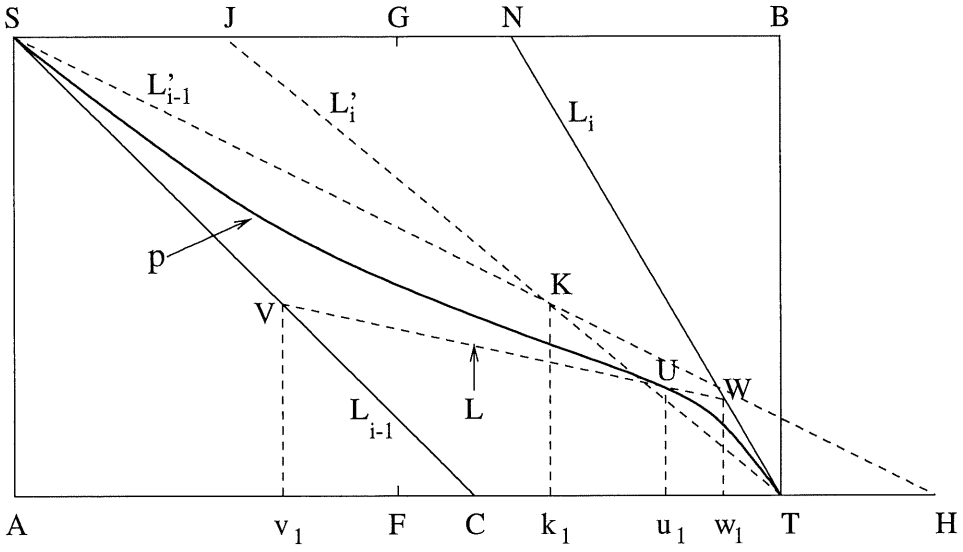


Figure 2.5 – *Méthode de McAllister et Roulier :* 3ème cas

(1) Soit H le point de la droite qui passe par A et T, et tel que C soit le milieu du segment $[A, H]$. On définit L'_{i-1} la droite qui passe par S et H.

(2) Soit J le point de la droite qui passe par B et S, et tel que N soit le milieu du segment $[B, J]$. On définit L'_i la droite qui passe par T et J (voir la Figure 2.5).

Soit $K = (k_1, k_2)$ le point d'intersection des droites L'_{i-1} et L'_i. On introduit un seul nœud additionnel U d'abscisse

$$u_1 = \xi_i = \frac{k_1 + x_i}{2}, \tag{2.14}$$

et on définit les points V, W, U et la spline quadratique p comme dans les deux cas précédents. Notons que dans ce cas, la spline quadratique doit changer de convexité sur l'intervalle $[x_{i-1}, x_i]$.

2.2.4 Quatrième cas

Dans ce cas, aucune des droites L_{i-1} et L_i n'intersecte le segment $\mathbf{M}$ dans $\mathcal{R}$ (voir la Figure 2.6).

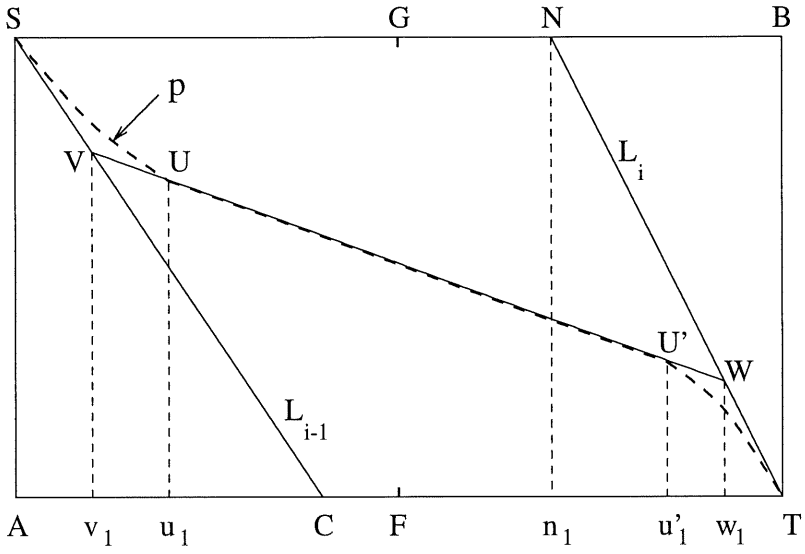


Figure 2.6 – *Méthode de McAllister et Roulier :* 4ème cas

C'est le seul cas qui nécessite l'ajout de deux nœuds dans l'intervalle $]x_{i-1}, x_i[$:

$$u_1 = \xi_i = \frac{c_1 + x_{i-1}}{2} \qquad \text{et} \qquad u_1' = \zeta_i = \frac{n_1 + x_i}{2}$$

où c_1 et n_1 sont les abscisses des points C et N respectivement. Posons

$$V = (v_1, v_2) = \left(\frac{x_{i-1} + \xi_i}{2}, L_{i-1}\left(\frac{x_{i-1} + \xi_i}{2}\right)\right)$$

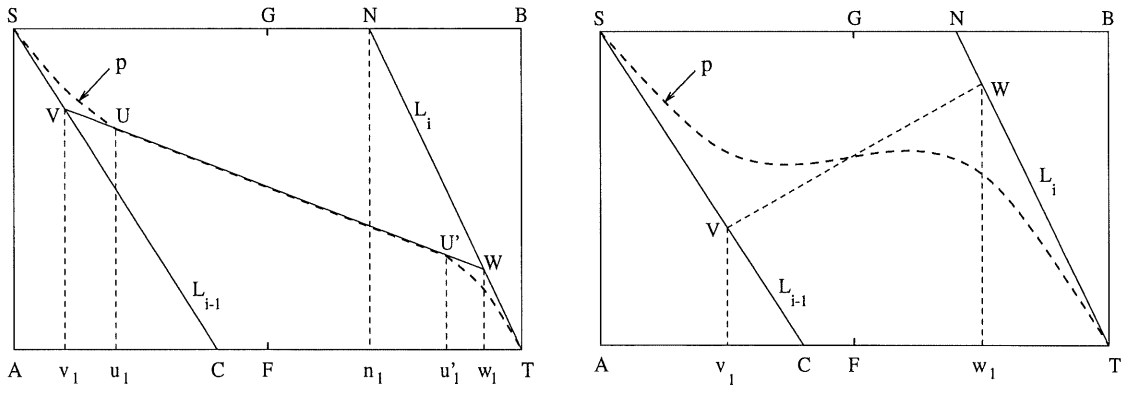
$$W = (w_1, w_2) = \left(\frac{x_i + \zeta_i}{2}, L_i\left(\frac{x_i + \zeta_i}{2}\right)\right)$$

On définit $\eta_i = L(\xi_i)$ et $\mu_i = L(\zeta_i)$, où L est la droite qui passe par V et W. La spline quadratique p est définie sur $[x_{i-1}, x_i]$ en utilisant les deux nœuds supplémentaires $U = (\xi_i, \eta_i)$ et $U' = (\zeta_i, \mu_i)$. Par conséquent, la spline quadratique p est partitionnée en trois

sections de polynômes quadratiques. Par construction, la section du milieu correspond à un polynôme de Bernstein linéaire, qui est une droite (voir la Figure 2.6). Comme dans le troisième cas, la spline quadratique doit changer de convexité sur l'intervalle $[x_{i-1}, x_i]$.

Remarque 2.2.2

1. *Le premier cas peut être exprimé analytiquement par l'expression suivante :* $(d_{i-1} - \delta_i)(d_i - \delta_i) < 0$, *qui correspond au premier cas de la méthode de Schumaker qui va être présentée dans le prochain chapitre.*
2. *Le nœud additionnel* ξ_i *n'est pas uniquement déterminé. En effet,* ξ_i *peut être choisi n'importe où dans* $]x_{i-1}, \bar{\xi}]$ *ou dans* $[\underline{\xi}, x_i[$ *en fonction du signe de* $|d_{i-1} - \delta_i| - |d_i - \delta_i|$. *On verra ce point plus en détail au Lemme 3.3.6 du prochain chapitre.*
3. *Les trois derniers cas peuvent être regroupés sous l'expression analytique suivante :* $(d_{i-1} - \delta_i)(d_i - \delta_i) \geq 0$, *qui correspond au deuxième cas de la méthode de Schumaker.*
4. *Dans les deuxième et troisième cas, le nœud additionnel* ξ_i *n'est pas uniquement déterminé. En effet,* ξ_i *peut être choisi n'importe où dans l'intervalle* $]x_{i-1}, x_i[$. *Toutefois, on risque de perdre la monotonie des données.*
5. *Dans le quatrième cas, on peut se contenter d'ajouter un seul nœud dans l'intervalle* $]x_{i-1}, x_i[$, *mais on risque de perdre la monotonie de la spline (voir la Figure 2.7).*



(a) Avec deux nœuds (b) Avec un seul nœud

Figure 2.7 – *Illustration de l'intérêt d'avoir deux nœuds*

2.3 Le choix des pentes

McAllister et Roulier [15] ont décrit une technique locale pour assigner les pentes aux données. Les pentes sont calculées de façon à se limiter aux deux premiers cas. Ainsi la spline quadratique résultante p aura exactement un nœud additionnel inséré entre chaque paire de données et préservera localement la monotonie et/ou la convexité des données. Le calcul de la pente en un point intérieur dépend de ses deux points avoisinants, tandis qu'aux extrémités (x_0, y_0) et (x_n, y_n), le calcul de la pente dépend des deux points qui le succèdent immédiatement (ou qui le précèdent).

Ainsi, aux points internes $(x_i, y_i), i = 1, \ldots, n-1$, si $\delta_i \delta_{i+1} \leq 0$, on prend $d_i = 0$. Sinon, lorsque $|\delta_i| > |\delta_{i+1}| > 0$, on prolonge la ligne de pente δ_i qui passe par (x_i, y_i) jusqu'à ce qu'elle intersecte la ligne horizontale passant par (x_{i+1}, y_{i+1}) au point $(\bar{x}, y_{i+1})$. Ensuite, on prend

$$\hat{x} = \frac{\bar{x} + x_{i+1}}{2}, \tag{2.15}$$

et on pose

$$d_i = \frac{y_{i+1} - y_i}{\hat{x} - x_i}. \tag{2.16}$$

D'autre part, lorsque $0 < |\delta_i| \leq |\delta_{i+1}|$, on fait l'inverse de la procédure précédente en prolongeant la ligne de pente δ_{i+1} passant par $S = (x_i, y_i)$ jusqu'à ce qu'elle intersecte la ligne horizontale passant par (x_{i-1}, y_{i-1}) au point $(\bar{x}, y_{i-1})$. Ensuite, on prend

$$\hat{x} = \frac{x_{i-1} + \bar{x}}{2}, \tag{2.17}$$

et on pose

$$d_i = \frac{y_i - y_{i-1}}{x_i - \hat{x}}. \tag{2.18}$$

Supposons maintenant que la pente d_{n-1} ait été déterminée. Au point extrême (x_n, y_n) la pente d_n est calculée comme suit :

(i) si $\delta_{n-1}\delta_n < 0$, on pose

$$d_n = 2\delta_n; \tag{2.19}$$

(ii) dans le cas contraire, on considère

$$\bar{x} = x_{n-\frac{1}{2}}, \tag{2.20}$$

on calcule

$$\bar{y} = d_{n-1}(\bar{x} - x_{n-1}) + y_{n-1} \tag{2.21}$$

et alors

$$d = \frac{y_n - \bar{y}}{x_n - \bar{x}} = 2\delta_n - d_{n-1}. \tag{2.22}$$

Finalement si $d\delta_n \leq 0$, alors on prend $d_n = 0$, sinon on prend $d_n = d$.

Ce qu'on peut résumer par

$$d_n = \begin{cases} 0, & \text{si } \delta_n(2\delta_n - d_{n-1}) \leq 0, \\ 2\delta_n - d_{n-1}, & \text{sinon.} \end{cases} \tag{2.23}$$

Note Si d_{n-1} est calculée en utilisant (2.15) et (2.16), alors $d\delta_n \geq 0$ et $d_n = d$.

Le calcul de la pente d_0 au point (x_0, y_0) se fait de façon similaire

$$d_0 = \begin{cases} 0, & \text{si } \delta_1(2\delta_1 - d_1) \leq 0, \\ 2\delta_1 - d_1, & \text{sinon.} \end{cases} \tag{2.24}$$

Proposition 2.3.1 *En adoptant ce choix de pentes, on obtient toujours le premier ou le second cas de la section précédente.*

Démonstration Soit $\mathcal{R}_i$ le rectangle déterminé par les points (x_{i-1}, y_{i-1}) et (x_i, y_i), $i = 1, \ldots, n$. L'objectif est de montrer que la droite de pente d_i qui passe par le point

(x_i, y_i) intersecte les segments médians des deux rectangles avoisinants $\mathcal{R}_i$ et $\mathcal{R}_{i+1}$. Dans le cas des points extrêmes ($i = 0$ ou $i = n$), on montre ce résultat pour le rectangle intérieur seulement.

Premièrement, dans le cas où $d_i = 0$, l'assertion est triviale puisque la droite de pente nulle passant par (x_i, y_i) forme une frontière horizontale avec chacun des rectangles avoisinants, et par suite, intersecte le segment médian de chaque rectangle. Donc il suffit de considérer les cas où $d_i \neq 0$.

Considérons le cas d'un point intérieur (x_i, y_i), avec $d_i > 0$ et $0 < \delta_{i+1} < \delta_i$ (pour les autres cas, on retrouve les mêmes résultats de façon similaire).

Dans ce cas, $(\bar{x}, y_{i+1})$ est le point d'intersection de la droite de pente δ_i passant par le point (x_i, y_i) et la droite horizontale passant par le point (x_{i+1}, y_{i+1}). L'équation de la droite passant par (x_i, y_i) de pente δ_i est donnée par

$$y - y_i = \delta_i(x - x_i). \tag{2.25}$$

Comme

$$y_{i+1} - y_i = \delta_i(\bar{x} - x_i)$$

on a

$$\bar{x} = x_i + \frac{y_{i+1} - y_i}{\delta_i} = x_i + \frac{\delta_{i+1}}{\delta_i}(x_{i+1} - x_i) \tag{2.26}$$

où $0 < \dfrac{\delta_{i+1}}{\delta_i} < 1$ par hypothèse. Aussi $\hat{x}$ est donné par

$$\begin{aligned} \hat{x} &= \frac{\bar{x} + x_{i+1}}{2} = \frac{x_i + x_{i+1}}{2} + \frac{y_{i+1} - y_i}{2\delta_i} \\ &= \frac{x_i + x_{i+1}}{2} + \frac{1}{2}\,\frac{\delta_{i+1}}{\delta_i}\,(x_{i+1} - x_i) \end{aligned} \tag{2.27}$$

d'où on tire

$$\frac{x_i + x_{i+1}}{2} < \hat{x} < x_{i+1}. \tag{2.28}$$

En utilisant (2.27) on a également

$$\begin{aligned} \hat{x} - x_i &= \frac{x_i + x_{i+1}}{2} + \frac{y_{i+1} - y_i}{2\delta_i} - x_i \\ &= \frac{x_{i+1} - x_i}{2} + \frac{y_{i+1} - y_i}{2\delta_i} \end{aligned} \tag{2.29}$$

et puisque

$$d_i = \frac{y_{i+1} - y_i}{\hat{x} - x_i} > 0,$$

on obtient

$$\frac{1}{d_i} = \frac{1}{2}\left(\frac{1}{\delta_i} + \frac{1}{\delta_{i+1}}\right). \tag{2.30}$$

À partir des hypothèses faites sur δ_i et δ_{i+1} on peut immédiatement obtenir

$$\delta_{i+1} < d_i < 2\delta_{i+1} \tag{2.31}$$

et

$$0 < d_i < \delta_i \tag{2.32}$$

La première inégalité (2.31) implique que la droite L_i de pente d_i passant par (x_i, y_i) intersecte le segment médian du rectangle $\mathcal{R}_{i+1}$. La seconde inégalité (2.32) implique que L_i intersecte le segment médian de $\mathcal{R}_i$.

En effet, l'équation de la droite passant par (x_i, y_i) ayant la pente d_i est donnée par :

$$y - y_i = d_i(x - x_i) \tag{2.33}$$

Soit $\bar{y}$ l'ordonnée au point $x_{i+\frac{1}{2}}$

$$\begin{aligned} \bar{y} &= y_i + d_i \left(x_{i+\frac{1}{2}} - x_i\right) \\ &= y_i + d_i \frac{x_{i+1} - x_i}{2}, \end{aligned} \tag{2.34}$$

et ainsi de (2.31)

$$y_i < \bar{y} < y_i + 2\delta_{i+1} \frac{x_{i+1} - x_i}{2} = y_{i+1}.$$

Donc $(x_{i+\frac{1}{2}}, \bar{y})$ est le point d'intersection de la droite L_i et du segment médian de $\mathcal{R}_{i+1}$. D'autre part, soit $\tilde{y}$ l'ordonnée au point $x_{i-\frac{1}{2}}$

$$\begin{aligned} \tilde{y} &= y_i + d_i \left(x_{i-\frac{1}{2}} - x_i \right) \\ &= y_i + d_i \frac{x_{i-1} - x_i}{2}, \end{aligned} \tag{2.35}$$

et donc, d'après (2.32)

$$y_{i-1} < \frac{y_{i-1} + y_i}{2} < y_i + \delta_i \frac{x_{i-1} - x_i}{2} < \tilde{y} < y_i$$

Donc $(x_{i-\frac{1}{2}}, \tilde{y})$ est le point d'intersection de la droite L_i et du segment médian de $\mathcal{R}_i$. Analysons maintenant le cas de l'extrémité (x_n, y_n) avec la pente $d_n > 0$. Ou bien d_n est donnée par (2.19), ou bien $d_n, \bar{x}$ et $\bar{y}$ sont données par (2.22), (2.20) et (2.21).
Dans le premier cas, la droite L_n de pente d_n passant par (x_n, y_n) intersecte la ligne médiane du rectangle $\mathcal{R}_{n-1}$. En effet, l'équation de L_n est donnée par :

$$y - y_n = d_n (x - x_n) = 2\delta_n (x - x_n) \tag{2.36}$$

donc pour $x = x_{n-\frac{1}{2}}$, on a $\tilde{y} = y_{n-1}$, d'où $(x_{n-\frac{1}{2}}, y_{n-1})$ est le point d'intersection de la droite L_n et du segment médian de $\mathcal{R}_n$. Dans le deuxième cas, l'équation de L_n est donnée par :

$$\begin{aligned} y - y_n &= d (x - x_n) \\ &= \frac{y_n - \bar{y}}{x_n - \bar{x}} (x - x_n). \end{aligned} \tag{2.37}$$

À partir de cette dernière équation, on peut facilement vérifier que le point $(\bar{x}, \bar{y})$ appartient à la droite L_n. C'est aussi le point où la droite L_{n-1} de pente d_{n-1} passant par (x_{n-1}, y_{n-1}) intersecte le segment médian du rectangle $\mathcal{R}_n$ (d'après (2.20) et (2.21)). Donc L_n aussi intersecte le segment médian.
De façon similaire, on peut montrer que la droite L_0 de pente d_0 passant par le point extrême (x_0, y_0) intersecte le segment médian du rectangle $\mathcal{R}_1$. ■

Remarque 2.3.2

1. *La pente d_i donnée par l'équation (2.30) est la moyenne harmonique de δ_i et δ_{i+1}.*
2. *En 1980, Butland [3] a été le premier à utiliser cette moyenne dans l'estimation des pentes. C'est la raison pour laquelle la moyenne harmonique est souvent appelée pente de Butland.*
3. *DeVore et Yan [6] ont remarqué que les pentes données par McAllister et Roulier, en utilisant des constructions géométriques compliquées, pouvaient avoir une description plus simple à l'aide des moyennes harmoniques. Ceci a été démontré explicitement par Iqbal [10].*

2.4 L'algorithme

Dans cette section la notation $[\delta, (x, y)]$ représente la droite de pente δ passant par le point (x, y)

$$[\delta, (x, y)] = \{(\xi, \eta) \mid \eta = y + \delta(\xi - x)\}$$

1. **Calcul des pentes :**

   Pour les points internes, où $i = 1, \ldots, n-1$:

   - Si $\delta_i \delta_{i+1} \leq 0$, alors $d_i = 0$.
   - Sinon,
     - Si $|\delta_i| > |\delta_{i+1}| > 0$,
       . On calcule $(\bar{x}, y_{i+1}) = [\delta_i, (x_i, y_i)] \cap [0, (x_{i+1}, y_{i+1})]$,
       . On prend $\hat{x} = \dfrac{\bar{x} + x_{i+1}}{2}$,
       . On calcule $d_i = \dfrac{y_{i+1} - y_i}{\hat{x} - x_i}$.
     - Si $0 < |\delta_i| \leq |\delta_{i+1}|$,
       . On calcule $(\bar{x}, y_{i-1}) = [\delta_{i+1}, (x_i, y_i)] \cap [0, (x_{i-1}, y_{i-1})]$,

. On prend $\hat{x} = \dfrac{x_{i-1} + \bar{x}}{2}$,

. On calcule $d_i = \dfrac{y_i - y_{i-1}}{x_i - \hat{x}}$.

Au point extrême (x_n, y_n) :

- Si $\delta_{n-1}\delta_n < 0$, alors $d_n = 2\delta_n$.
- Sinon,
  - . $\bar{x} = x_{n-\frac{1}{2}}$,
  - . $\bar{y} = d_{n-1}(\bar{x} - x_{n-1}) + y_{n-1}$,
  - . $d = \dfrac{y_n - \bar{y}}{x_n - \bar{x}}$,
  - . Si $d\delta_n \leq 0$, alors $d_n = 0$,
  - . Sinon $d_n = d$.

Au point extrême (x_0, y_0) :

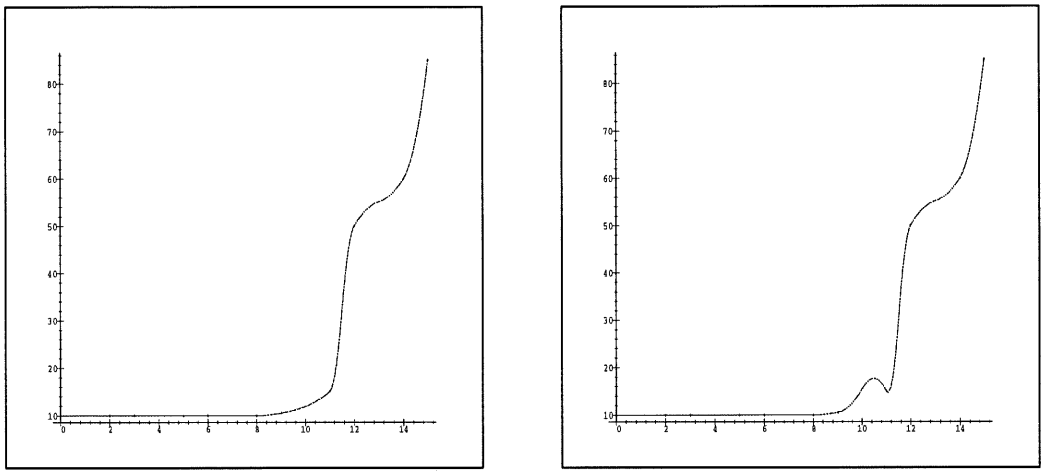
- Si $\delta_1\delta_2 < 0$, alors $d_0 = 2\delta_1$.
- Sinon,
  - . $\bar{x} = x_{\frac{1}{2}}$,
  - . $\bar{y} = d_1(\bar{x} - x_1) + y_1$,
  - . $d = \dfrac{y_0 - \bar{y}}{x_0 - \bar{x}}$,
  - . Si $d\delta_1 \leq 0$, alors $d_0 = 0$,
  - . Sinon $d_0 = d$.

2. **Calcul des nœuds et des coefficients :**

   Voir la première annexe.

2.5 Conclusion

1. La méthode de McAllister et Roulier est une méthode "locale". Pour illustrer ceci, la pente au point $(11, 15)$ de l'exemple d'`Akima` est remplacée par -1. La spline résultante est représentée à la Figure 2.8b. On remarque que les deux courbes sont identiques sauf au voisinage du point $(11, 15)$.



(a) (b)

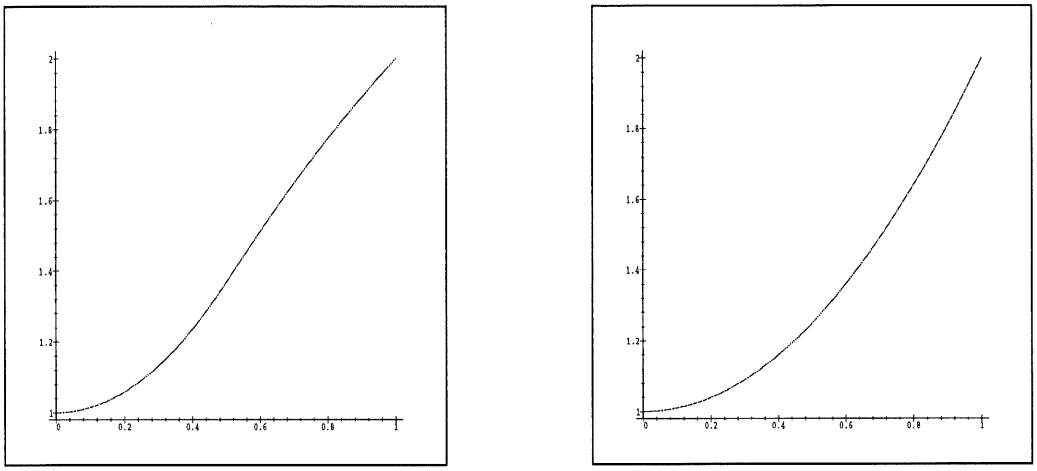
Figure 2.8 – *Illustration du caractère local de la méthode*

2. L'algorithme présente un phénomène *pathologique* dans le sens où un petit changement des pentes peut causer une variation radicale dans la spline résultante. On est face à une telle instabilité lorsque la pente d_i au point (x_i, y_i) est presque égale à la pente δ_i (ou δ_{i+1}). Par exemple :
   Soient

$$(x_{i-1}, y_{i-1}) = (0, 1), \qquad (x_i, y_i) = (1, 2), \qquad d_{i-1} = 0.0, \qquad d_i = 1.05,$$

   alors $\delta_i = 1$. Ainsi l'algorithme génère le premier cas, qui produit la spline donnée à la Figure 2.9a. Cependant, si d_i est modifiée à 0.95, l'algorithme génère le deuxième

cas, qui produit la spline donnée à la Figure 2.9b.



(a) Cas $d_i = 1.05$ (b) Cas $d_i = 0.95$

Figure 2.9 – *Illustration du phénomène pathologique*

Le problème ici est dû au fait que le signe de $d_i - \delta_i$ détermine la présence ou non d'un point d'inflexion sur $[x_{i-1}, x_i]$. Tout algorithme local qui préserve la monotonie et la convexité doit faire face à ce problème.

Pour remédier à ce problème, McAllister et Roulier [16] ont introduit dans leur algorithme, une constante de tolérance ε, qui permet de choisir le deuxième cas au lieu du premier quand la différence entre δ_i et d_i est très petite ($|d_i - \delta_i| < \varepsilon$).

CHAPITRE 3

ALGORITHME DE SCHUMAKER

3.1 Introduction

En 1983, Schumaker [17] a décrit une méthode assez simple qui préserve la forme des données, il construit la fonction interpolante comme une fonction quadratique de classe $\mathcal{C}^1$ où les nœuds sont les données $x_0, \ldots, x_n$, avec au plus un nœud additionnel sur chaque sous-intervalle $[x_{i-1}, x_i]$, où $i = 1, \ldots, n$.

Les nœuds additionnels sont insérés sous certaines conditions, de façon à ce que la spline résultante préserve la forme des données, et avec une certaine liberté pour choisir où les placer exactement.

La méthode est basée sur un nombre de lemmes qui caractérisent la solution du problème d'interpolation d'Hermite (1.5).

3.2 Cas $d_{i-1} + d_i = 2\delta_i$

Dans ce cas, les lemmes démontrés par Schumaker [17] nous permettent de caractériser la solution du problème d'interpolation d'Hermite (1.5).

3.2.1 Existence de la solution

Lemme 3.2.1 *Il existe un polynôme de second degré solution du problème d'interpolation d'Hermite (1.5) sur* $[x_{i-1}, x_i]$ *si et seulement si* $d_{i-1} + d_i = 2\delta_i$. *Dans ce cas le polynôme s'écrit sous la forme*

$$p(x) = y_{i-1} + d_{i-1}(x - x_{i-1}) + \frac{d_i - d_{i-1}}{x_i - x_{i-1}} \frac{(x - x_{i-1})^2}{2}, \qquad x \in [x_{i-1}, x_i] \tag{3.1}$$

Démonstration On construit p tel que $p(x_{i-1}) = y_{i-1}$, $p'(x_{i-1}) = d_{i-1}$ et $p(x_i) = y_i$ en utilisant

$$p(x) = y_{i-1}H_{i-1}(x) + d_{i-1}\hat{H}_{i-1}(x) + y_iH_i(x)$$

où

$$H_{i-1}(x) = \frac{x_i - x}{x_i - x_{i-1}}\left[1 + \frac{x - x_{i-1}}{x_i - x_{i-1}}\right], \hat{H}_{i-1}(x) = \frac{(x - x_{i-1})(x_i - x)}{x_i - x_{i-1}} \text{ et } H_i(x) = \frac{(x - x_{i-1})^2}{(x_i - x_{i-1})^2}$$

possèdent les propriétés suivantes

$$H_j(x_k) = \delta_{jk}, \qquad H'_j(x_{i-1}) = 0,$$

$$\hat{H}_{i-1}(x_j) = 0, \qquad \hat{H}'_{i-1}(x_{i-1}) = 1,$$

où $j = i - 1, i$ et $k = i - 1, i$.

On vérifie alors directement (3.1) et $p'(x_i) = d_i$. ■

3.2.2 Monotonie et convexité

Lemme 3.2.2 *Si $d_{i-1}+d_i = 2\delta_i$, alors l'unique polynôme quadratique p sur $[x_{i-1}, x_i]$ est monotone si et seulement si d_{i-1} et d_i sont de même signe. De plus p est convexe (resp. concave) sur $[x_{i-1}, x_i]$ si $d_i \geq d_{i-1}$ (resp. $d_i \leq d_{i-1}$).*

Démonstration On a

$$\begin{aligned} p'(x) &= d_{i-1} + \frac{d_i - d_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}), \qquad x \in [x_{i-1}, x_i] \\ &= \frac{x_i - x}{x_i - x_{i-1}} d_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} d_i \end{aligned}$$

donc

$$p'(x) \geq 0, \quad \forall x \in [x_{i-1}, x_i] \quad \text{si et seulement si} \quad d_i, d_{i-1} \geq 0$$

et

$$p'(x) \leq 0, \quad \forall x \in [x_{i-1}, x_i] \quad \text{si et seulement si} \quad d_i, d_{i-1} \leq 0$$

ce qui prouve l'assertion de la monotonie.

D'autre part, on a

$$p''(x) = \frac{d_i - d_{i-1}}{x_i - x_{i-1}}, \qquad x \in [x_{i-1}, x_i]$$

donc

$$p''(x) \geq 0 \quad \forall x \in [x_{i-1}, x_i] \quad \text{si et seulement si} \quad d_i \geq d_{i-1}$$

et

$$p''(x) \leq 0 \quad \forall x \in [x_{i-1}, x_i] \quad \text{si et seulement si} \quad d_i \leq d_{i-1}$$

ce qui prouve l'assertion de la convexité. ■

Remarque 3.2.3 *Notons que si $d_{i-1} + d_i = 2\delta_i$, les droites*

$$\begin{cases} y &= y_{i-1} + d_{i-1}(x - x_{i-1}) \\ y &= y_i + d_i(x - x_i) \end{cases}$$

sont confondues si $d_{i-1} = d_i$. Dans le cas contraire, les droites se rencontrent en $x = x_{i-\frac{1}{2}}$.

Par conséquent, le polynôme de second degré, solution du problème d'interpolation d'Hermite, donné par (3.1) au Lemme 3.2.1 coïncide avec le polynôme de Bernstein de degré deux donné par la Proposition 2.1.1.

3.3 Cas $d_{i-1} + d_i \neq 2\delta_i$

Dans ce cas, en ajoutant un nœud intermédiaire, il est possible de déterminer une spline quadratique solution du problème d'interpolation d'Hermite.

3.3.1 Existence de la solution

Lemme 3.3.1 *Lorsque $d_{i-1} + d_i \neq 2\delta_i$, en ajoutant un nœud additionnel $\xi_i \in]x_{i-1}, x_i[$, il existe $p \in C^1[x_{i-1}, x_i]$ telle que*

$$p(x) = \begin{cases} A_1 + B_1(x - x_{i-1}) + C_1(x - x_{i-1})^2, & x \in [x_{i-1}, \xi_i] \\ \\ A_2 + B_2(x - \xi_i) + C_2(x - \xi_i)^2, & x \in [\xi_i, x_i] \end{cases} \tag{3.2}$$

avec

$$A_1 = y_{i-1}, \qquad B_1 = d_{i-1}, \qquad C_1 = \frac{\bar{d}_i - d_{i-1}}{2\alpha}$$

$$A_2 = y_{i-1} + d_{i-1}\alpha + \frac{(\bar{d}_i - d_{i-1})\alpha}{2}, \qquad B_2 = \bar{d}_i, \qquad C_2 = \frac{d_i - \bar{d}_i}{2\beta}$$

où

$$\bar{d}_i = p'(\xi_i) = 2\delta_i - [\omega d_{i-1} + (1 - \omega)d_i] \tag{3.3}$$

et

$$\alpha = \xi_i - x_{i-1}, \qquad \beta = x_i - \xi_i, \qquad \omega = \frac{\alpha}{x_i - x_{i-1}}, \qquad 1 - \omega = \frac{\beta}{x_i - x_{i-1}}. \tag{3.4}$$

Démonstration En utilisant le Lemme 3.2.1 avec un nœud intermédiaire ξ_i, nous avons sur $[x_{i-1}, \xi_i]$

$$p(x) = A_1 + B_1(x - x_{i-1}) + C_1(x - x_{i-1})^2$$

avec

$$A_1 = y_{i-1}, \qquad B_1 = d_{i-1}, \qquad C_1 = \frac{\bar{d}_i - d_{i-1}}{2(\xi_i - x_{i-1})}$$

et

$$d_{i-1} + \bar{d}_i = 2\delta_i^0 = 2\,\frac{p(\xi_i) - p(x_{i-1})}{\xi_i - x_{i-1}}.$$

Donc p est telle que

$$p(x_{i-1}) = y_{i-1}, \qquad p'(x_{i-1}) = d_{i-1},$$

$$p(\xi_i) = p(\xi_i), \qquad p'(\xi_i) = \bar{d}_i.$$

De même, sur $[\xi_i, x_i]$ on a

$$p(x) = A_2 + B_2(x - \xi_i) + C_2(x - \xi_i)^2$$

avec

$$A_2 = p(\xi_i) = A_1 + B_1(\xi_i - x_{i-1}) + C_1(\xi_i - x_{i-1})^2,$$

$$B_2 = \bar{d}_i, \quad C_2 = \frac{d_i - \bar{d}_i}{2(x_i - \xi_i)}$$

et

$$\bar{d}_i + d_i = 2\delta_i^1 = 2\,\frac{p(x_i) - p(\xi_i)}{x_i - \xi_i}.$$

Donc p est telle que

$$p(\xi_i) = p(\xi_i), \qquad p'(\xi_i) = \bar{d}_i,$$

$$p(x_i) = p(x_i), \qquad p'(x_i) = d_i.$$

Il reste à choisir $\bar{d}_i$ de telle sorte que $p(x_i) = y_i$. Ainsi

$$\begin{aligned} y_i &= p(x_i) = A_2 + B_2(x_i - \xi_i) + C_2(x_i - \xi_i)^2 \\ &= [A_1 + B_1(\xi_i - x_{i-1}) + C_1(\xi_i - x_{i-1})^2] + B_2(x_i - \xi_i) + C_2(x_i - \xi_i)^2 \\ &= \left[y_{i-1} + d_{i-1}(\xi_i - x_{i-1}) + \frac{\bar{d}_i - d_{i-1}}{2(\xi_i - x_{i-1})}(\xi_i - x_{i-1})^2\right] + \bar{d}_i(x_i - \xi_i) + \frac{d_i - \bar{d}_i}{2(x_i - \xi_i)}(x_i - \xi_i)^2 \\ &= y_{i-1} + \frac{\bar{d}_i + d_{i-1}}{2}(\xi_i - x_{i-1}) + \frac{\bar{d}_i + d_i}{2}(x_i - \xi_i), \end{aligned}$$

d'où

$$\begin{aligned} \bar{d}_i &= 2\delta_i - \frac{\alpha d_{i-1} + \beta d_i}{x_i - x_{i-1}} \\ &= 2\delta_i - [\omega d_{i-1} + (1-\omega)d_i] \end{aligned}$$

avec $\omega = \dfrac{\xi_i - x_{i-1}}{x_i - x_{i-1}}$ et $1 - \omega = \dfrac{x_i - \xi_i}{x_i - x_{i-1}}$.

Finalement, pour un ξ_i fixé, $\bar{d}_i$ est uniquement déterminée par y_{i-1}, y_i, d_{i-1} et d_i, d'où l'unicité de la solution. ■

3.3.2 Monotonie

Schumaker [17] a présenté les résultats suivants sur la forme de la fonction interpolante.

Lemme 3.3.2 *p est monotone sur $[x_{i-1}, x_i]$ si et seulement si d_{i-1}, d_i et $\bar{d}_i$ sont de même signe.*
De plus, comme $\bar{d}_i = 2\delta_i - [\omega d_{i-1} + (1-\omega)d_i]$, lorsque

$$\begin{cases} d_{i-1}, d_i \geq 0 \quad \textit{il faut et il suffit que} \quad \omega d_{i-1} + (1-\omega)d_i \leq 2\delta_i, \\ \\ d_{i-1}, d_i \leq 0 \quad \textit{il faut et il suffit que} \quad \omega d_{i-1} + (1-\omega)d_i \geq 2\delta_i. \end{cases} \tag{3.5}$$

Démonstration On a

$$p'(x) = \begin{cases} d_{i-1}\dfrac{\xi_i - x}{\xi_i - x_{i-1}} + \bar{d}_i\dfrac{x - x_{i-1}}{\xi_i - x_{i-1}}, & x \in [x_{i-1}, \xi_i]; \\ \bar{d}_i\dfrac{x_i - x}{x_i - \xi_i} + d_i\dfrac{x - \xi_i}{x_i - \xi_i}, & x \in [\xi_i, x_i]. \end{cases} \tag{3.6}$$

Comme p est monotone sur $[x_{i-1}, x_i]$ si et seulement si p' a toujours le même signe sur cet intervalle, on en déduit le résultat. ■

Lemme 3.3.3 *Lorsque d_{i-1}, d_i et δ_i sont de même signe, la solution du problème d'interpolation d'Hermite est monotone sur $[x_{i-1}, x_i]$ si et seulement si*

$$\min(|d_{i-1}|, |d_i|) \begin{cases} < 2|\delta_i|, & si\ d_{i-1} \neq d_i, \\ \leq 2|\delta_i|, & si\ d_{i-1} = d_i, \end{cases} \tag{3.7}$$

et le nœud additionnel ξ_i est choisi tel que

$$\begin{cases} \max(x_{i-1}, c_i) < \xi_i < x_i, & si\ |d_{i-1}| < |d_i|, \\ x_{i-1} < \xi_i < x_i, & si\ |d_{i-1}| = |d_i|, \\ x_{i-1} < \xi_i < \min(x_i, c_i), & si\ |d_{i-1}| > |d_i|, \end{cases} \tag{3.8}$$

où le point c_i est donné par l'expression

$$c_i = x_{i-1} + \frac{(x_i - x_{i-1})(d_i - 2\delta_i)}{d_i - d_{i-1}} = x_i - \frac{(x_i - x_{i-1})(2\delta_i - d_{i-1})}{d_i - d_{i-1}}.$$

Démonstration La condition (3.5) du Lemme 3.3.2, qui est nécessaire et suffisante pour avoir la monotonie, peut être exprimée par

$$2|\delta_i| \geq \frac{1}{x_i - x_{i-1}} [\, (\xi_i - x_{i-1})|d_{i-1}| + (x_i - \xi_i)|d_i| \,]. \tag{3.9}$$

À partir de cette dernière inégalité, on peut extraire une condition pour placer le nœud additionnel ξ_i, en fonction des valeurs $x_{i-1}, x_i, d_{i-1}, d_i$ et δ_i qui sont les données du problème. D'autre part, il faut tenir compte du fait que ξ_i doit satisfaire

$$x_{i-1} < \xi_i < x_i. \tag{3.10}$$

Si $|d_{i-1}| = |d_i|$, il est facile de voir que (3.9) est vérifiée pour tout $\xi_i \in]x_{i-1}, x_i[$ si et seulement si $|d_{i-1}| \leq 2|\delta_i|$.

Supposons alors que $|d_{i-1}| < |d_i|$, dans ce cas (3.9) est équivalente à la condition

$$\xi_i \geq x_{i-1} + \frac{(x_i - x_{i-1})(d_i - 2\delta_i)}{d_i - d_{i-1}} = c_i.$$

De plus, la condition (3.10) exige que

$$c_i < x_i. \tag{3.11}$$

Il est facile de voir que (3.11) est vérifiée si les pentes d_{i-1} et d_i respectent la condition (3.7). Ainsi (3.9) et (3.10) sont vérifiées pour tous les nœuds $\xi_i \in]\max(x_{i-1}, c_i), x_i[$. Ce qui prouve le lemme pour les valeurs $|d_{i-1}| < |d_i|$.

Le cas $|d_{i-1}| > |d_i|$ peut être traité de façon similaire. ■

3.3.3 Convexité

Lemme 3.3.4

- *p est convexe sur $[x_{i-1}, x_i]$ si et seulement si $d_{i-1} \leq \bar{d}_i \leq d_i$;*
- *p est concave sur $[x_{i-1}, x_i]$ si et seulement si $d_i \leq \bar{d}_i \leq d_{i-1}$.*

Démonstration Comme

$$p''(x) = \begin{cases} \dfrac{\bar{d}_i - d_{i-1}}{\alpha}, & x \in [x_{i-1}, \xi_i], \\ \dfrac{d_i - \bar{d}_i}{\beta}, & x \in [\xi_i, x_i], \end{cases}$$

alors p est convexe si et seulement si $\bar{d}_i - d_{i-1} \geq 0$ et $d_i - \bar{d}_i \geq 0$,

et p est concave si et seulement si $\bar{d}_i - d_{i-1} \leq 0$ et $d_i - \bar{d}_i \leq 0$. ■

Lemme 3.3.5 *Si $(d_i - \delta_i)(d_{i-1} - \delta_i) \geq 0$ alors p possède un point d'inflexion en ξ_i.*

Démonstration On a

$$\bar{d}_i - d_{i-1} = (1+\omega)(\delta_i - d_{i-1}) + (1-\omega)(\delta_i - d_i)$$

et

$$\bar{d}_i - d_i = \omega(\delta_i - d_{i-1}) + (2-\omega)(\delta_i - d_i)$$

Si $(d_i - \delta_i)(d_{i-1} - \delta_i) \geq 0$ alors $\bar{d}_i - d_{i-1}$ et $\bar{d}_i - d_i$ sont de même signe (au moins un des deux est non nul, sinon $d_{i-1} + d_i = 2\delta_i$), et donc p a un point d'inflexion en ξ_i. ■

Lemme 3.3.6

$$\textit{Si } (d_i - \delta_i)(d_{i-1} - \delta_i) < 0 \qquad \textit{alors} \qquad \begin{cases} p \textit{ est convexe si} & d_{i-1} < d_i \\ \\ p \textit{ est concave si} & d_{i-1} > d_i \end{cases} \tag{3.12}$$

et ξ_i est choisi tel que

$$x_i - 2(x_i - x_{i-1})\frac{\delta_i - d_{i-1}}{d_i - d_{i-1}} \leq \xi_i < x_i \quad si\ |d_{i-1} - \delta_i| < |d_i - \delta_i| \tag{3.13}$$

$$x_{i-1} < \xi_i \leq x_{i-1} + 2(x_i - x_{i-1})\frac{d_i - \delta_i}{d_i - d_{i-1}} \quad si\ |d_{i-1} - \delta_i| > |d_i - \delta_i| \tag{3.14}$$

Dans les autres cas il y a un point d'inflexion.

Démonstration On introduit

$$\begin{aligned} g_{i-1}(\xi_i) &= \bar{d}_i - d_{i-1} = 2\delta_i - [\omega d_{i-1} + (1-\omega)d_i] - d_{i-1} \\ &= (1+\omega)(\delta_i - d_{i-1}) + (1-\omega)(\delta_i - d_i) \\ &= [(\delta_i - d_{i-1}) + (\delta_i - d_i)] + \omega[(\delta_i - d_{i-1}) - (\delta_i - d_i)] \\ &= -[(d_i - \delta_i) - (\delta_i - d_{i-1})] + \omega(d_i - d_{i-1}) \\ &= 2\delta_i - (d_{i-1} + d_i) + \omega(d_i - d_{i-1}) \end{aligned} \tag{3.15}$$

et

$$\begin{aligned} g_i(\xi_i) &= d_i - \bar{d}_i = d_i - \{2\delta_i - [\omega d_{i-1} + (1-\omega)d_i]\} \\ &= (d_i - \delta_i) + \omega(d_{i-1} - \delta_i) + (1-\omega)(d_i - \delta_i) \\ &= (d_i - \delta_i) + (d_{i-1} - \delta_i) + (1-\omega)[(d_i - \delta_i) - (d_{i-1} - \delta_i)] \\ &= [(d_i - \delta_i) - (\delta_i - d_{i-1})] + (1-\omega)(d_i - d_{i-1}) \\ &= -[2\delta_i - (d_{i-1} + d_i)] + (1-\omega)(d_i - d_{i-1}) \end{aligned} \tag{3.16}$$

Si $(d_i - \delta_i)(d_{i-1} - \delta_i) < 0$, on est dans l'une des situations suivantes :

(i) $d_{i-1} < \delta_i < d_i$ (et donc $d_{i-1} < d_i$ qui est une condition nécessaire à la convexité)

(ii) $d_i < \delta_i < d_{i-1}$ (et donc $d_i < d_{i-1}$ qui est une condition nécessaire à la concavité)

Pour (i)

si $0 < \delta_i - d_{i-1} < d_i - \delta_i$ alors $g_i(\xi_i) \geq 0$ pour tout ξ_i
et $g_{i-1}(\xi_i) \geq 0$ si et seulement si $\xi_i \geq \underline{\xi} = x_i - 2\dfrac{\delta_i - d_{i-1}}{d_i - d_{i-1}}(x_i - x_{i-1})$
si $0 < d_i - \delta_i < \delta_i - d_{i-1}$ alors $g_{i-1}(\xi_i) \geq 0$ pour tout ξ_i
et $g_i(\xi_i) \geq 0$ si et seulement si $\xi_i \leq \bar{\xi} = x_{i-1} + 2\dfrac{d_i - \delta_i}{d_i - d_{i-1}}(x_i - x_{i-1})$

Pour (ii)

si $d_i - \delta_i < \delta_i - d_{i-1} < 0$ alors $g_i(\xi_i) \leq 0$ pour tout ξ_i
et $g_{i-1}(\xi_i) \leq 0$ si et seulement si $\xi_i \geq \underline{\xi} = x_i - 2\dfrac{\delta_i - d_{i-1}}{d_i - d_{i-1}}(x_i - x_{i-1})$
si $\delta_i - d_{i-1} < d_i - \delta_i < 0$ alors $g_{i-1}(\xi_i) \leq 0$ pour tout ξ_i
et $g_i(\xi_i) \leq 0$ si et seulement si $\xi_i \leq \bar{\xi} = x_{i-1} + 2\dfrac{d_i - \delta_i}{d_i - d_{i-1}}(x_i - x_{i-1})$

Comme

$$p''(x) = \begin{cases} \dfrac{\bar{d}_i - d_{i-1}}{\alpha} = \dfrac{g_{i-1}(\xi_i)}{\alpha}, & x \in [x_{i-1}, \xi_i] \\ \dfrac{d_i - \bar{d}_i}{\beta} = \dfrac{g_i(\xi_i)}{\beta}, & x \in [\xi_i, x_i] \end{cases}$$

le résultat est immédiat. ■

3.3.4 Commentaires

1. La condition d_{i-1}, d_i *et* δ_i *sont de même signe* est une condition nécessaire pour avoir la monotonie de la spline, tandis que le Lemme 3.3.2 nous donne une condition suffisante.

2. Le Lemme 3.3.4 nous donne une condition nécessaire et suffisante pour avoir la convexité ou la concavité de la spline.

3. Même si le Lemme 3.3.1 montre qu'on peut résoudre le problème d'interpolation d'Hermite par une spline quadratique avec un nœud arbitrairement placé dans l'intervalle I, on ne peut pas toujours satisfaire les conditions du Lemme 3.3.4. Le Lemme 3.3.6 nous montre exactement où il faut placer les nœuds pour avoir la convexité ou la concavité de la spline.

4. Le Lemme 3.3.6 nous donne des intervalles $([x_{i-1}, \bar{\xi}], [\underline{\xi}, x_i])$ où on peut choisir les nœuds afin de préserver la convexité (ou la concavité) des données et c'est dans ce sens qu'on parle d'une certaine *liberté* dans le choix des nœuds. Ces intervalles sont appelés *régions d'admissibilité.*

5. L'intervalle d'admissibilité pour la convexité est toujours contenu dans celui de la monotonie pourvu que $d_{i-1}d_i \geq 0$. En effet, si le nœud est choisi dans l'intervalle d'admissibilité pour la convexité et si d_{i-1} et d_i sont de même signe, alors d'après le Lemme 3.3.4, $\bar{d}_i$ possède le même signe que d_{i-1} et d_i. Donc la fonction linéaire p' ne change pas de signe sur $[x_{i-1}, x_i]$, et par suite p est monotone sur cet intervalle.

3.4 Le choix des nœuds

Schumaker [17] a proposé de choisir les nœuds au centre des régions d'admissibilité, ainsi :

Cas 1 *Si* $(d_i - \delta_i)(d_{i-1} - \delta_i) < 0$, *alors :*

$$\xi_i = x_i - \frac{(\delta_i - d_{i-1})(x_i - x_{i-1})}{d_i - d_{i-1}} \quad quand \quad |d_{i-1} - \delta_i| < |d_i - \delta_i| \tag{3.17}$$

$$\xi_i = x_{i-1} + \frac{(d_i - \delta_i)(x_i - x_{i-1})}{d_i - d_{i-1}} \quad quand \quad |d_{i-1} - \delta_i| > |d_i - \delta_i| \tag{3.18}$$

Cas 2 *Si* $(d_i - \delta_i)(d_{i-1} - \delta_i) \geq 0$, *alors :*

$$\xi_i = x_{i-\frac{1}{2}} \tag{3.19}$$

3.5 Le choix des pentes

Pour les points $\{(x_i, y_i)\}_{i=0}^n$ on définit

$$L_i = [(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2]^{1/2}, \qquad i = 1, \ldots, n \tag{3.20}$$

L_i mesure la *longueur* de la corde reliant les deux points successifs (x_{i-1}, y_{i-1}) et (x_i, y_i), alors que δ_i mesure la *pente* de cette corde. Ainsi contrairement à δ_i, L_i est une quantité toujours positive.

Au point (x_i, y_i), on définit la quantité $\tilde{L}_i$ comme étant la somme de L_i et de toutes les longueurs avoisinantes ayant la même pente δ_i.

$$\tilde{L}_i = \sum_{l_i}^{r_i} L_j, \qquad i = 1, \ldots, n \tag{3.21}$$

où l_i et r_i sont les derniers indices (à gauche et à droite) pour lesquels les pentes sont égales.

$$\delta_{l_i - 1} \neq \delta_{l_i} = \ldots = \delta_i = \ldots = \delta_{r_i} \neq \delta_{r_i + 1}$$

Schumaker a proposé de calculer les pentes de la façon suivante :

La pente d_i au point (x_i, y_i) est calculée comme étant une moyenne pondérée de δ_i et δ_{i+1}, où les poids sont les quantités $\tilde{L}_i$ et $\tilde{L}_{i+1}$

$$d_i = \frac{\tilde{L}_i \delta_i + \tilde{L}_{i+1} \delta_{i+1}}{\tilde{L}_i + \tilde{L}_{i+1}}, \qquad i = 1, \ldots, n-1 \tag{3.22}$$

En particulier, si $\delta_i = \delta_{i+1}$, alors $d_i = \delta_i = \delta_{i+1}$. La pente d_0 en (x_0, y_0) est calculée à partir de δ_1 et d_1 de la façon suivante : $d_0 = \frac{1}{2}(3\delta_1 - d_1)$. De façon similaire, la pente d_n en (x_n, y_n) est donnée par : $d_n = \frac{1}{2}(3\delta_n - d_{n-1})$.

3.6 L'algorithme

1. **Initialisation :**

   - Pour $i = 1, \ldots, n$

     . $L_i = [(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2]^{1/2}$

     . $\tilde{L}_i = \sum_{l_i}^{r_i} L_j$, où : $\delta_{l_i - 1} \neq \delta_{l_i} = \ldots = \delta_{r_i} \neq \delta_{r_i + 1}$

2. **Calcul des pentes :**

   - Pour $i = 1, \ldots, n-1$

     . $d_i = \dfrac{\tilde{L}_i \delta_i + \tilde{L}_{i+1} \delta_{i+1}}{\tilde{L}_i + \tilde{L}_{i+1}}$

   - $d_0 = (3\delta_1 - d_1)/2$

   - $d_n = (3\delta_n - d_{n-1})/2$

3. **Calcul des nœuds et des coefficients :**

   - $j = 0$

   - Pour $i = 1, \ldots, n$

     • si $d_{i-1} + d_i = 2\delta_i$

       $j = j + 1, x_j = x_{i-1}, A_j = y_{i-1}, B_j = d_{i-1}$

$$C_j = \frac{d_i - d_{i-1}}{2(x_i - x_{i-1})}$$

- sinon
  - si $(d_{i-1} - \delta_i)(d_i - \delta_i) \geq 0$

    $\xi_i = x_{i-\frac{1}{2}}$
  - sinon
    - si $|d_{i-1} - \delta_i| < |d_i - \delta_i|$

      $$\xi_i = x_i - \frac{(\delta_i - d_{i-1})(x_i - x_{i-1})}{d_i - d_{i-1}}$$
    - sinon

      $$\xi_i = x_{i-1} + \frac{(d_i - \delta_i)(x_i - x_{i-1})}{d_i - d_{i-1}}$$

    $\alpha = \xi_i - x_{i-1}, \beta = x_i - \xi_i$

    $$\bar{d}_i = 2\delta_i - \frac{\alpha d_{i-1} + \beta d_i}{x_i - x_{i-1}}$$

    $j = j + 1, x_j = x_{i-1}, A_j = y_{i-1}, B_j = d_{i-1}$

    $$C_j = \frac{\bar{d}_i - d_{i-1}}{2\alpha}$$

    $j = j + 1, x_j = \xi_i, A_j = y_{i-1} + d_{i-1}\alpha + (\bar{d}_i - d_{i-1})\alpha/2$

    $$B_j = \bar{d}_i, C_j = \frac{d_i - \bar{d}_i}{2\beta}$$

Pour la codification de cet algorithme, voir la deuxième annexe.

3.7 L'algorithme interactif

Le choix des nœuds, dans les intervalles où il est nécessaire de les introduire, se fait de façon à assurer la monotonie locale et la convexité (ou la concavité) locale. Cependant, il se peut qu'en assurant la convexité locale, la condition (3.5) qui assure la monotonie soit violée (voir la Figure 3.2a). C'est pour cette raison, que Schumaker a introduit son `algorithme interactif` (voir la Figure 3.1), qui permet à l'utilisateur d'ajuster la position des nœuds, et/ou de modifier les pentes.

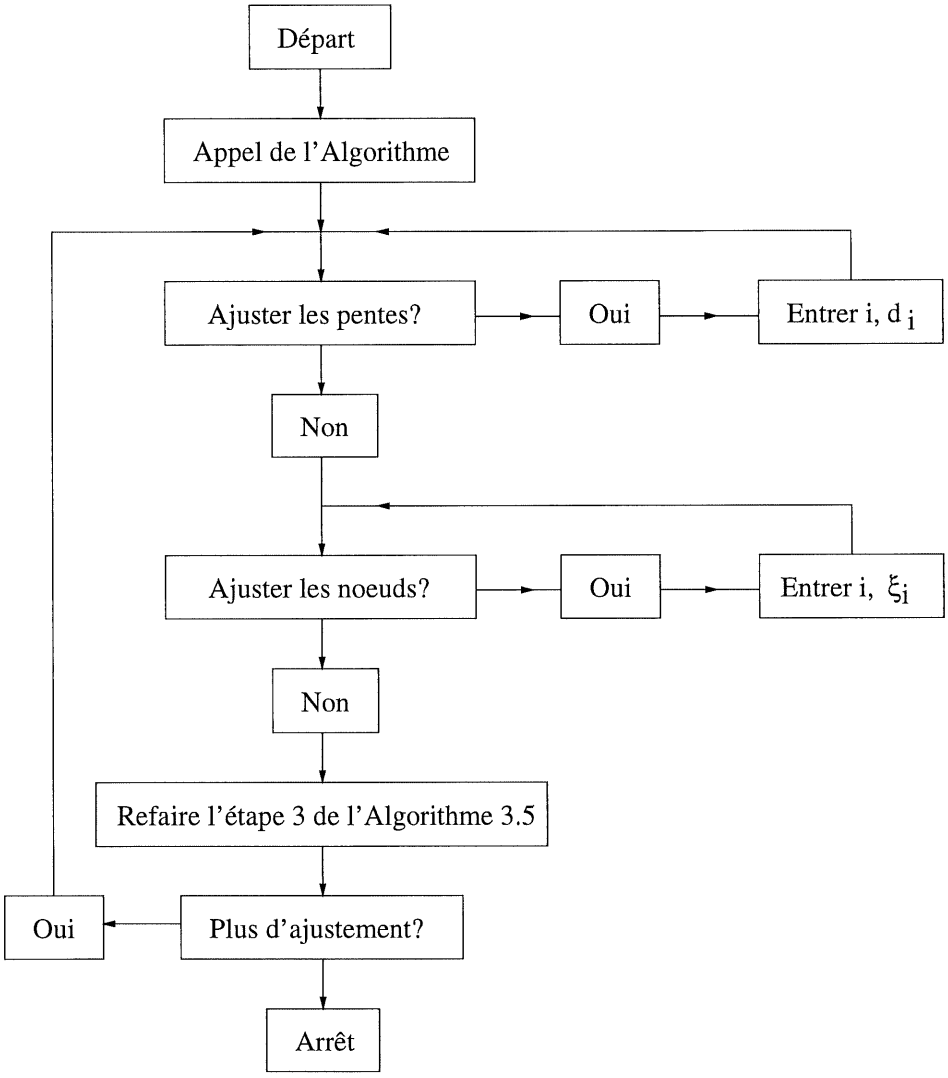


Figure 3.1 – *Illustration de l'algorithme interactif de Schumaker*

3.7.1 Ajustement des pentes

L'utilisateur est libre d'ajuster les pentes à n'importe quel point, mais il doit tenir compte de la relation entre la valeur de δ_i et la forme de p :

- Une condition nécessaire pour avoir la monotonie sur $[x_{i-1}, x_i]$ est $d_{i-1}d_i \geq 0$.
- Une condition nécessaire pour avoir la monotonie sur $[x_i, x_{i+1}]$ est $d_i d_{i+1} \geq 0$.
- Une condition nécessaire pour avoir la monotonie à la fois sur $[x_{i-1}, x_i]$ et $[x_i, x_{i+1}]$, dans le cas où $\delta_i \delta_{i+1} \leq 0$, est $d_i = 0$.
- La condition (3.5) est nécessaire pour assurer la monotonie sur un intervalle où p a un point d'inflexion.

3.7.2 Ajustement des nœuds

L'utilisateur est aussi libre d'ajuster la position d'un nœud ξ_i dans chaque intervalle où il est nécessaire d'en avoir un. En choisissant les nœuds, il doit tenir compte de ceci :

- ξ_i peut être choisi n'importe où dans $]x_{i-1}, x_i[$ quand $a_i b_i \geq 0$, où $a_i = d_{i-1} - \delta_i$ et $b_i = d_i - \delta_i$.
- ξ_i peut être choisi n'importe où dans $]x_{i-1}, \bar{\xi}]$ quand $a_i b_i < 0$, et $|a_i| > |b_i|$.
- ξ_i peut être choisi n'importe où dans $[\underline{\xi}, x_i[$ quand $a_i b_i < 0$, et $|a_i| < |b_i|$.

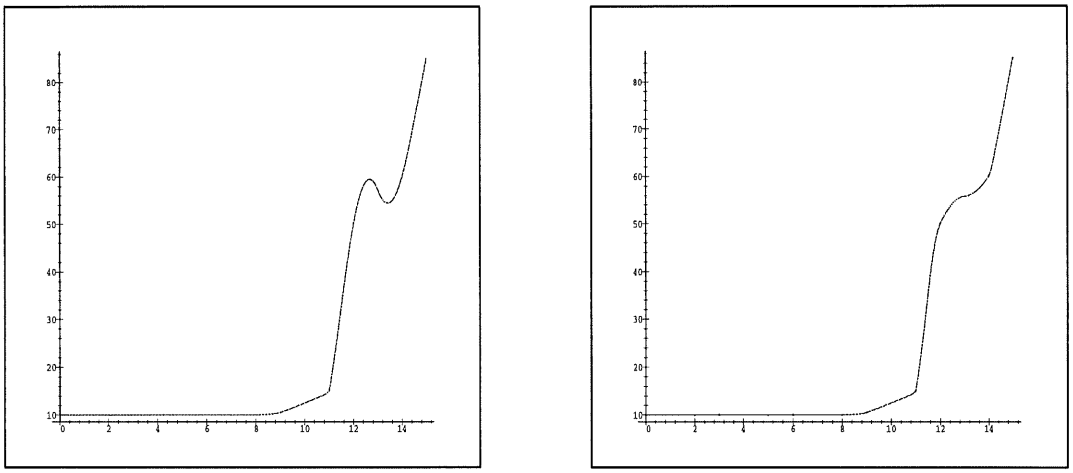
3.8 Conclusion

1. L'algorithme interactif de Schumaker, comme son nom l'indique, permet à l'utilisateur d'ajuster les pentes et/ou les nœuds jusqu'à ce que la courbe obtenue soit satisfaisante (préserve la forme des données). La Figure 3.2 illustre l'application de

cet algorithme sur l'exemple d'`Akima`. La première application de l'algorithme 3.6 produit la spline représentée par la Figure 3.2a. Les pentes choisies par cet algorithme sont données par le tableau suivant :

Tableau 3.1 – *Les pentes sélectionnées par l'Algorithme 3.6*

x_i	0	2	3	5	6	8	9	11	12	14	15
d_i	0	0	0	0	0	0.061	1.92	30.96	28.23	19.21	27.85



(a) Première application de l'Algorithme 3.6

(b) Après ajustement des pentes

Figure 3.2 – *Application de l'algorithme interactif sur l'exemple d'*`Akima`

L'algorithme 3.6 considère les points $\{7, 8.76, 10.977, 11.5, 13, 14.33\}$ comme nœuds supplémentaires. On remarque que même si les données sont partout croissantes, la spline interpolante ne l'est pas sur l'intervalle $[12, 14]$.
Dans la Figure 3.2b, les pentes aux points 12 et 14 sont, respectivement, remplacées par 11 et 8. Ce changement nous assure la monotonie de la spline sur $[12, 14]$.

2. Contrairement à la méthode de McAllister et Roulier, qui est une méthode géométrique, la méthode de Schumaker a l'avantage d'être une méthode analytique, donc très facile à programmer. Cependant, le choix de pentes fait par Schumaker n'est pas optimal. En effet, on verra ultérieurement, qu'en adoptant un autre choix de pentes, on obtient de meilleurs résultats, dans le sens où on trouve la solution optimale après une seule exécution de l'algorithme sans faire appel à l'algorithme interactif.
3. On peut traiter le problème d'interpolation d'Hermite par la méthode de Schumaker sans mettre en évidence le premier cas, $d_{i-1} + d_i = 2\delta_i$.
4. Enfin, la méthode de Schumaker, à l'instar des méthodes qui utilisent des splines quadratiques, est une méthode *locale*. Donc, en ajustant un nœud ou une pente, la courbe ne change qu'au voisinage du point où la valeur a été modifiée.

CHAPITRE 4

LE CHOIX DE PENTES DE LAHTINEN

4.1 Introduction

Le choix des pentes représente une étape très importante pour les algorithmes qui préservent la forme des données. Étant donné que les quantités δ_i et δ_{i+1} donnent des informations sur la forme des données localement au point (x_i, y_i), l'idée est de choisir la pente en ce point comme une combinaison de ces deux valeurs.
Ainsi, comme on a vu au deuxième chapitre, McAllister et Roulier ont choisi la pente d_i comme étant une moyenne harmonique des pentes δ_i et δ_{i+1}

$$\frac{1}{d_i} = \frac{1}{2}\left(\frac{1}{\delta_i} + \frac{1}{\delta_{i+1}}\right),$$

alors que Schumaker a choisi la pente d_i comme une combinaison convexe de δ_i et δ_{i+1}

$$d_i = \frac{\tilde{L}_i\delta_i + \tilde{L}_{i+1}\delta_{i+1}}{\tilde{L}_i + \tilde{L}_{i+1}},$$

où les $\tilde{L}_i$ sont donnés par les formules (3.20) et (3.21).

Dans ce chapitre, on va voir que le choix de Lahtinen est une généralisation du choix de Schumaker, ensuite on présentera quelques résultats obtenus par Lahtinen [11][12][13].

4.2 Le choix de Lahtinen

Au point (x_i, y_i), Lahtinen a choisi la pente de la façon suivante

$$d_i = \begin{cases} a_i\delta_i + (1-a_i)\delta_{i+1}, \quad 0 \le a_i \le 1, & \text{si } \delta_i\delta_{i+1} > 0, \\ 0, & \text{si } \delta_i\delta_{i+1} \le 0, \end{cases} \tag{4.1}$$

avec $i = 1, \ldots, n-1$, et les pentes aux extremités comme suit

$$d_0 = a_0\delta_1, \quad a_0 \ge 0, \qquad\qquad d_n = a_n\delta_n, \quad a_n \ge 0. \tag{4.2}$$

La pente d_i est choisie comme étant une combinaison convexe de δ_i et δ_{i+1}, cependant si ces dernières valeurs sont de signes opposés ou que l'une d'entre elles au moins est nulle, on prend $d_i = 0$. Ce dernier choix est nécessaire pour assurer la monotonie des données. Évidemment ceci implique que lorsque $\delta_i = 0$, on a $d_{i-1} = d_i = 0$ et la solution du problème d'interpolation d'Hermite est constante sur l'intervalle $[x_{i-1}, x_i]$.

Remarque 4.2.1

1. *Dans la formule (4.1), si $\delta_i\delta_{i+1} \le 0$, alors il existe un unique $a_i \in [0,1]$ tel que*

$$d_i = a_i\delta_i + (1-a_i)\delta_{i+1} = 0. \tag{4.3}$$

   En effet, si $\delta_i\delta_{i+1} = 0$, alors $a_i = 1$ ou 0 respectivement. Et si $\delta_i\delta_{i+1} < 0$, alors il suffit de prendre $a_i = \dfrac{\delta_{i+1}}{\delta_{i+1} - \delta_i}$, pour avoir (4.3).

2. *Les pentes de Schumaker sont données par $d_i = \dfrac{\tilde{L}_i\delta_i + \tilde{L}_{i+1}\delta_{i+1}}{\tilde{L}_i + \tilde{L}_{i+1}}$, où les $\tilde{L}_i$ sont donnés par les formules (3.20) et (3.21). En choisissant le paramètre*

$$a_{Si} = \frac{\tilde{L}_i}{\tilde{L}_i + \tilde{L}_{i+1}},$$

on voit bien que le choix de pentes de Schumaker est un cas particulier du choix de Lahtinen.

3. *Les pentes de McAllister et Roulier sont de la forme (4.1) avec* $a_{Mi} = \dfrac{\delta_{i+1}}{\delta_i + \delta_{i+1}}$.

Dans la formule (4.1) les nombres a_i sont des paramètres. Dans la suite de ce chapitre, on s'interesse aux pentes qui préservent la forme des données pour un choix particulier des paramètres a_i.

4.3 La monotonie

Soit p la solution du problème d'interpolation d'Hermite dans le cas où les pentes sont données par (4.1) et (4.2). On commence par examiner comment choisir les valeurs des paramètres a_i pour que p préserve la monotonie des données. Il est suffisant de prouver la monotonie localement, ainsi on considère la situation dans l'intervalle $[x_{i-1}, x_i]$. La condition (3.7) du Lemme 3.3.3, qui restreint les pentes, est essentielle. Si elle est satisfaite, on peut toujours choisir le nœud additionnel en respectant (3.8).

4.3.1 Le choix des paramètres a_i

Proposition 4.3.1 *Soit p une solution du problème d'interpolation d'Hermite sur l'intervalle $[x_{i-1}, x_i]$, où les pentes d_{i-1} et d_i sont données par (4.1). Alors il existe des intervalles A_j, $j = i-1, i$, et B_i tels que p préserve la monotonie des données pour chaque $a_{i-1} \in A_{i-1}, a_i \in A_i$ et $\xi_i \in B_i$, s'il est nécessaire d'ajouter un nœud.*

Démonstration Pour un intervalle $[x_{i-1}, x_i]$, $i = 2, \ldots, n-1$

(1) Si les pentes δ_{i-1}, δ_i et δ_{i+1} ne sont pas de même signe, alors au moins une des dérivées d_{i-1} ou d_i est nulle, et dans ce cas (3.7) est vérifiée et p préserve la monotonie des

données pour n'importe quel choix des paramètres

$$0 \leq a_{i-1} \leq 1, \qquad 0 \leq a_i \leq 1. \tag{4.4}$$

(2) Si les pentes δ_{i-1}, δ_i et δ_{i+1} sont de même signe, d'après (4.1), on a

$$\min(|\delta_j|, |\delta_{j+1}|) \leq |d_j| \leq \max(|\delta_j|, |\delta_{j+1}|), \qquad j = i-1, i.$$

ce qui implique que

$$\min(|d_{i-1}|, |d_i|) \leq \min[\max(|\delta_{i-1}|, |\delta_i|), \max(|\delta_i|, |\delta_{i+1}|)]$$

(i) Si

$$\min(|\delta_{i-1}|, |\delta_{i+1}|) < 2|\delta_i|, \tag{4.5}$$

d'après (3.7), la spline p préserve la monotonie des données pour des paramètres a_i comme dans (4.4).

(ii) Sinon, p ne préserve la monotonie des données que dans l'une des situations suivantes

$$0 \leq a_{i-1} < \frac{\delta_i}{\delta_{i-1} - \delta_i}, \qquad 0 \leq a_i \leq 1, \tag{4.6}$$

ou

$$0 \leq a_{i-1} \leq 1, \qquad 1 - \frac{\delta_i}{\delta_{i+1} - \delta_i} < a_i \leq 1. \tag{4.7}$$

Dans le premier cas on a $|d_{i-1}| < 2|\delta_i|$, et dans le second $|d_i| < 2|\delta_i|$. Dans les deux cas, on a $\min(|d_{i-1}|, |d_i|) < 2|\delta_i|$ qui est une condition suffisante pour avoir la monotonie de p, d'après le Lemme 3.3.3.

Ainsi pour un intervalle $[x_{i-1}, x_i]$, $i = 2, \ldots, n-1$, on peut toujours trouver des valeurs a_{i-1}, a_i et un nœud additionnel, si nécessaire, de façon à ce que la solution p du problème d'interpolation d'Hermite préserve la monotonie des données sur cet intervalle.

Le résultat est toujours vrai pour les intervalles $[x_0, x_1]$ et $[x_{n-1}, x_n]$. Considérons par exemple le premier intervalle $[x_0, x_1]$

(1) Si δ_1 et δ_2 sont de signe différent ou que $|\delta_2| < 2|\delta_1|$, alors $d_1 = 0$ ou $|d_1| < 2|\delta_1|$ et par suite $\min(|d_0|, |d_1|) < 2|\delta_1|$. D'où p préserve la monotonie des données pour

$$a_0 \in \mathbb{R}_+, \qquad 0 \leq a_1 \leq 1. \tag{4.8}$$

(2) Si δ_1 et δ_2 sont de même signe et que $|\delta_2| \geq 2|\delta_1|$, alors p préserve la monotonie des données dans l'une des situations suivantes

$$a_0 \in \mathbb{R}_+, \qquad 1 - \frac{\delta_1}{\delta_2 - \delta_1} < a_1 \leq 1. \tag{4.9}$$

ou

$$0 \leq a_0 < 2, \qquad 0 \leq a_1 \leq 1. \tag{4.10}$$

Dans le premier cas on a $|d_1| < 2|\delta_1|$, et dans le second $|d_0| < 2|\delta_1|$. Dans les deux cas, on a $\min(|d_0|, |d_1|) < 2|\delta_1|$ qui est une condition suffisante pour avoir la monotonie de p, d'après le Lemme 3.3.3.

On obtient des résultats similaires sur la préservation de la monotonie des données sur l'intervalle $[x_{n-1}, x_n]$. ■

4.3.2 Discussion

(1) Pour le choix de pentes de McAllister et Roulier, où

$$d_i = \begin{cases} \dfrac{2\delta_i\delta_{i+1}}{\delta_i + \delta_{i+1}}, & \text{si } \delta_i\delta_{i+1} > 0, \\ 0, & \text{si } \delta_i\delta_{i+1} \leq 0, \end{cases}$$

on a toujours $|d_{i-1}| \leq 2|\delta_i|$ et $|d_i| \leq 2|\delta_i|$. Donc les pentes vérifient toujours les conditions (3.7) du Lemme 3.3.3, et par conséquent la solution p du problème d'interpolation d'Hermite est monotone pour tout nœud additionnel $\xi_i \in]x_{i-1}, x_i[$.

(2) Par contre, Schumaker [17] a remarqué que son choix des dérivées ne permettait pas toujours de préserver la monotonie des données. Afin de remédier à ce problème, Schumaker a eu recours à l'algorithme interactif (Section 3.7). Lahtinen a présenté une proposition ([13], prop. 2) qui donne une condition nécessaire et suffisante pour que la spline quadratique interpolante p, avec le choix de pentes de Schumaker et les nœuds additionnels appropriés, préserve la monotonie des données.

(3) En 1992, en comparant ces deux dernières méthodes, Iqbal [10] a remarqué qu'en utilisant les pentes de Butland avec l'algorithme de Schumaker, ce dernier génère automatiquement une fonction interpolante qui préserve la forme des données (sans avoir recours à l'algorithme interactif pour préserver la monotonie). Ceci est dû au fait que les pentes de Butland préservent la monotonie des données (voir le premier point de cette discussion).

4.4 La convexité

Proposition 4.4.1 *On suppose que les données sont strictement convexes (resp. strictement concaves) sur l'intervalle* $[x_{i-1}, x_i]$, $i = 2, \ldots, n-1$

$$\delta_{i-1} < \delta_i < \delta_{i+1} \quad (\textit{resp. } \delta_{i-1} > \delta_i > \delta_{i+1}),$$

et que p est une solution du problème d'interpolation d'Hermite avec les pentes d_{i-1} et d_i telles que la condition

$$0 < a_{i-1} < 1, \qquad 0 < a_i < 1, \tag{4.11}$$

soit vérifiée.
Si $\delta_i \neq 0$, alors p est convexe (resp. concave) sur $[x_{i-1}, x_i]$, quand le nœud additionnel ξ_i est choisi comme dans le Lemme 3.3.6, s'il est nécessaire d'en ajouter un.
Si $\delta_i = 0$, alors p est constante sur $[x_{i-1}, x_i]$.

Démonstration On suppose que les données sont strictement convexes sur l'intervalle $[x_{i-1}, x_i]$, $i = 2, \ldots, n-1$, alors $\delta_{i-1} < \delta_i < \delta_{i+1}$. Si la condition (4.11) est vérifiée, on a

$$\delta_{i-1} < d_{i-1} < \delta_i < d_i < \delta_{i+1} \tag{4.12}$$

En effet,

(1) si les pentes sont de même signe, il découle de la définition de d_i donnée par (4.1)

$\delta_{i-1} < d_{i-1} < \delta_i < d_i < \delta_{i+1}$

(2) si les pentes sont de signe opposé, on risque d'avoir l'une des situations suivantes

(i) $\delta_{i-1} < 0 < \delta_i < \delta_{i+1}$, et d'après (4.1) on a $d_{i-1} = 0$, et par suite

$\delta_{i-1} < 0 < \delta_i < d_i < \delta_{i+1}$

(ii) $\delta_{i-1} < \delta_i < 0 < \delta_{i+1}$, et d'après (4.1) on a $d_i = 0$, et par suite

$\delta_{i-1} < d_{i-1} < \delta_i < 0 < \delta_{i+1}$

Donc dans tous les cas, on a (4.12). Et d'après le Lemme 3.3.6, la solution du problème d'interpolation d'Hermite est convexe sur l'intervalle $[x_{i-1}, x_i]$.

Des résultats similaires sont obtenus pour la concavité. ■

La convexité locale sur chaque intervalle $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, implique la convexité globale sur l'intervalle $[a, b]$. En revanche, si les données sont globalement convexes, alors la Proposition 4.4.1 assure la convexité de p sur tout intervalle $[x_{i-1}, x_i]$, $i = 2, \ldots, n-1$. Ainsi il reste juste à vérifier si p se comporte de la même façon sur les intervalles $[x_0, x_1]$ et $[x_{n-1}, x_n]$.

Il est évident que si $a_1 \in]0, 1[$ et a_0 est choisit tel que

$$\begin{cases} (a_0 - 1)\delta_1(\delta_2 - \delta_1) < 0, & \text{si } \delta_1(\delta_2 - \delta_1) \neq 0, \\ a_0 = 1, & \text{si } \delta_1(\delta_2 - \delta_1) = 0, \end{cases} \tag{4.13}$$

alors p respecte la convexité sur $[x_0, x_1]$.
De même pour l'intervalle $[x_{n-1}, x_n]$, si $0 < a_{n-1} < 1$ et a_n est choisit tel que

$$\begin{cases} (a_n - 1)\delta_n(\delta_n - \delta_{n-1}) < 0, & \text{si } \delta_n(\delta_n - \delta_{n-1}) \neq 0, \\ a_n = 1, & \text{si } \delta_n(\delta_n - \delta_{n-1}) = 0, \end{cases} \tag{4.14}$$

alors p respecte la convexité sur $[x_{n-1}, x_n]$.

Supposons que les données sont globalement convexes et que p est la solution du problème d'interpolation d'Hermite, alors p est convexe sur $[a, b]$ si $0 < a_i < 1$, $2 \leq i \leq n-1$, a_0, a_n sont comme dans (4.13), (4.14) et le nœud additionnel ξ_i est choisit comme dans le Lemme 3.3.6.

Remarque 4.4.2 *Dans le choix de pentes de Schumaker, les paramètres a_{Si} sont tels que $0 < a_{Si} < 1$ pour tout i, et d'après la Proposition 4.4.1, la spline p associée à ce choix de pentes préserve la convexité des données.*

4.4.1 Point d'inflexion

Si les données ne sont ni globalement convexes ni globalement concaves, alors il faut étudier la forme de p sur les intervalles où la convexité des données change.

Proposition 4.4.3 *Soit p une solution du problème d'interpolation d'Hermite sur l'intervalle $[x_{i-1}, x_i]$. Si $d_{i-1} < \delta_i > d_i$ ou $d_{i-1} > \delta_i < d_i$, alors p a un nœud additionel ξ_i qui est aussi un point d'inflexion. Si $d_{i-1} < \delta_i > d_i$, alors p est convexe sur $[x_{i-1}, \xi_i]$ et concave sur $[\xi_i, x_i]$. Si $d_{i-1} > \delta_i < d_i$, alors p est concave sur $[x_{i-1}, \xi_i]$ et convexe sur $[\xi_i, x_i]$.*

Démonstration Voir la preuve du Lemme 3.3.5. ■

4.4.2 Discussion

(1) Supposons que les données sont convexes sur $[x_{i-2}, x_{i-1}]$ et concaves sur $[x_i, x_{i+1}]$, c'est-à-dire que $\delta_{i-2} < \delta_{i-1} < \delta_i > \delta_{i+1} > \delta_{i+2}$.

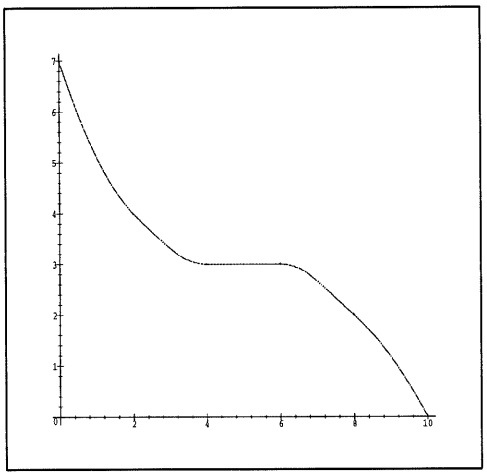
Tableau 4.1 – *Exemple de changement de convexité*

x_i	0	2	4	6	8	10
y_i	7	4	3	3	2	0

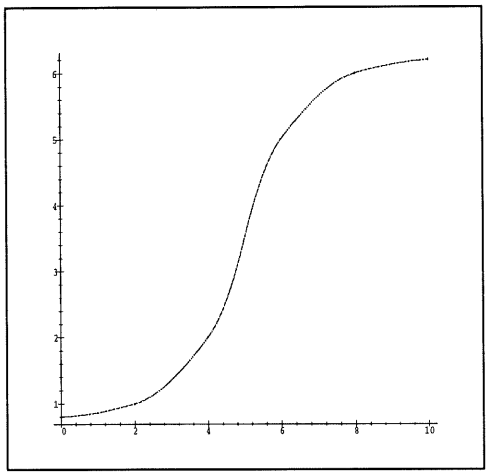
(a) $\delta_i = 0$

x_i	0	2	4	6	8	10
y_i	0.8	1	2	5	6	6.2

(b) $\delta_i \neq 0$



(a) $\delta_i = 0$



(b) $\delta_i \neq 0$

Figure 4.1 – *Exemple de changement de convexité*

Soit p la solution du problème d'interpolation d'Hermite, la Proposition 4.4.3 montre que pour tout $a_{i-1} \in]0,1[$ et $a_i \in]0,1[$, la spline est ou bien constante sur $[x_{i-1}, x_i]$ (voir la Figure 4.1a), ou bien elle possède un nœud additionnel ξ_i sur $[x_{i-1}, x_i]$ et dans ce cas p est convexe sur $[x_{i-2}, \xi_i]$ et concave sur $[\xi_i, x_{i+1}]$ (voir la Figure 4.1b).

(2) Si deux pentes adjacentes sont de même valeur, c'est-à dire $\delta_i = \delta_{i+1}$ alors la méthode présente un phénomène *pathologique*, dans le sens où la spline interpolante risque de ne pas respecter la forme des données. En effet, en supposant p la solution du problème d'interpolation d'Hermite où les dérivées sont données par (4.1) et que l'égalité des pentes survient entre deux régions de convexité : $\delta_{i-1} < \delta_i = \delta_{i+1} < \delta_{i+2}$,

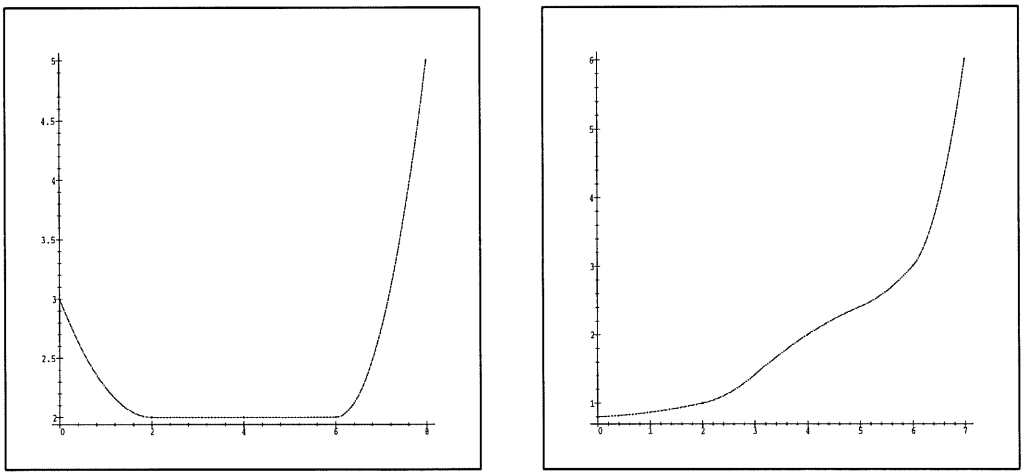
Tableau 4.2 – $\delta_i = \delta_{i+1}$ *entre deux régions de convexité*

x_i	0	2	4	6	8
y_i	3	2	2	2	5

(a) $\delta_i = \delta_{i+1} = 0$

x_i	0	2	4	6	7
y_i	0.8	1	2	3	6

(b) $\delta_i = \delta_{i+1} \neq 0$



(a) $\delta_i = \delta_{i+1} = 0$ (b) $\delta_i = \delta_{i+1} \neq 0$

Figure 4.2 – $\delta_i = \delta_{i+1}$ *entre deux régions de convexité*

alors on a

- si $\delta_i = 0$, p est constante sur $[x_{i-1}, x_{i+1}]$ (voir Figure 4.2a),

- sinon p a deux nœuds additionnels $\xi_i \in]x_{i-1}, x_i[$ et $\xi_{i+1} \in]x_i, x_{i+1}[$, qui sont aussi des points d'inflexion. Ainsi, p est concave sur $[\xi_i, \xi_{i+1}]$ et convexe ailleurs (voir Figure 4.2b). Des résultats similaires sont obtenus pour l'égalité des pentes entre deux régions de concavité.

(3) Si l'égalité des pentes survient entre une région de convexité et une région de concavité, tel que $\delta_{i-1} < \delta_i = \delta_{i+1} > \delta_{i+2}$,

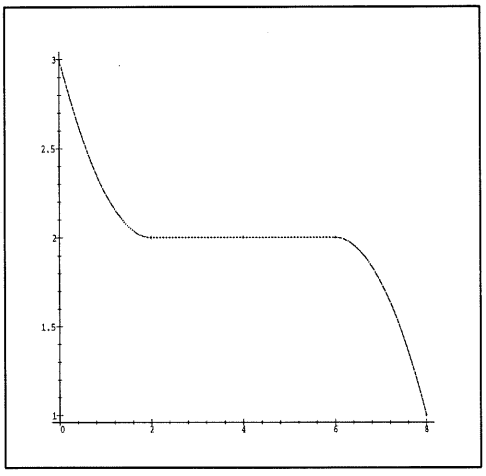
Tableau 4.3 – $\delta_i = \delta_{i+1}$ *entre une région de convexité et une région de concavité*

x_i	0	2	4	6	8
y_i	3	2	2	2	1

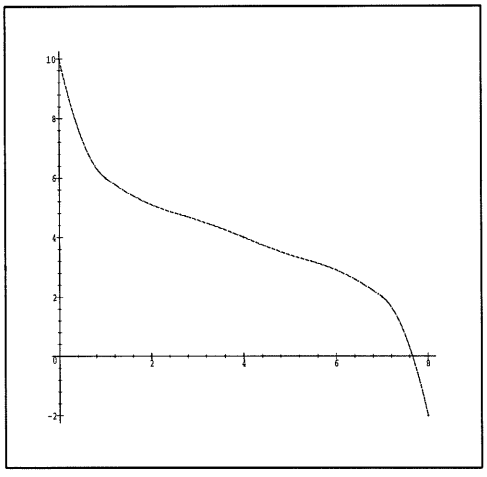
(a) $\delta_i = \delta_{i+1} = 0$

x_i	0	1	4	7	8
y_i	10	6	4	2	-2

(b) $\delta_i = \delta_{i+1} \neq 0$



(a) $\delta_i = \delta_{i+1} = 0$



(b) $\delta_i = \delta_{i+1} \neq 0$

Figure 4.3 – $\delta_i = \delta_{i+1}$ *entre une région de convexité et une région de concavité*

alors on a

- si $\delta_i = 0$, p est constante sur $[x_{i-1}, x_{i+1}]$ (voir Figure 4.3a),

- sinon p a deux nœuds additionnels $\xi_i \in]x_{i-1}, x_i[$ et $\xi_{i+1} \in]x_i, x_{i+1}[$, qui sont aussi des points d'inflexion. Et dans ce cas, x_i est aussi un point d'inflexion. Ainsi, p change de convexité trois fois sur l'intervalle $[x_{i-1}, x_{i+1}]$ (voir Figure 4.3b).

Remarque 4.4.4 *Sur l'intervalle $[x_{i-1}, x_{i+1}]$, les données sont linéaires, donc à la fois convexes et concaves. Par suite la fonction p solution du problème d'interpolation d'Hermite, bien qu'elle change de convexité, préserve localement la forme des données sur cet intervalle (ceci n'est plus vrai de façon globale).*

4.5 Conclusion

Étant donné $\{(x_i, y_i)\}_{i=0}^n$, si on choisit les pentes $\{d_i\}_{i=0}^n$ comme dans (4.1) et (4.2), alors il existe des intervalles $\{A_i\}_{i=0}^n$ et $\{B_i\}$ tels que la solution p du problème d'interpolation d'Hermite préserve la forme des données pour chaque $a_i \in A_i$ et chaque nœud additionnel $\xi_i \in B_i$, quand c'est nécessaire d'en ajouter un.

CHAPITRE 5

AMÉLIORATION DE L'ORDRE DE CONVERGENCE DES ALGORITHMES

5.1 Introduction

Le but de ce chapitre est d'améliorer l'ordre de convergence des algorithmes précédents. À cette fin, on suppose que les données $\{(x_i, y_i)\}_{i=0}^{n}$ proviennent d'une fonction régulière $f : [a, b] \longrightarrow \mathbb{R}$ avec $y_i = f(x_i)$. Ensuite, on analyse l'erreur $f - p$, où p est la spline quadratique solution du problème d'interpolation d'Hermite.

Sachant que l'algorithme de McAllister et Roulier est d'ordre $O(h^3)$ excepté au voisinage des racines de f', où il est seulement d'ordre $O(h^2)$, DeVore a eu l'idée de modifier la spline interpolante p quand f' est très petite afin d'améliorer la convergence.

Ce chapitre est en grande partie basé sur l'article de DeVore [6], dans lequel ce dernier introduit deux nouveaux algorithmes qui ont de meilleures propriétés de convergence.

5.2 L'algorithme de McAllister et Roulier

Soit h_i la moyenne harmonique de δ_i et δ_{i+1}

$$h_i = \frac{2\delta_i\delta_{i+1}}{\delta_i + \delta_{i+1}}, \qquad i = 1, \ldots, n-1.$$

Les pentes de l'algorithme de McAllister et Roulier sont données par

$$d_i = \begin{cases} 0, & \text{si } \delta_i\delta_{i+1} \leq 0 \\ h_i, & \text{si } \delta_i\delta_{i+1} > 0 \end{cases} \qquad \text{pour } i = 1, \ldots, n-1, \tag{5.1}$$

et aux extrémités de l'intervalle par

$$\begin{aligned} d_0 &= \begin{cases} 0, & \text{si } \delta_1(2\delta_1 - d_1) \leq 0, \\ 2\delta_1 - d_1, & \text{sinon,} \end{cases} \\ d_n &= \begin{cases} 0, & \text{si } \delta_n(2\delta_n - d_{n-1}) \leq 0, \\ 2\delta_n - d_{n-1}, & \text{sinon.} \end{cases} \end{aligned} \tag{5.2}$$

On a déjà remarqué que l'intervalle d'admissibilité pour la monotonie est toujours l'intervalle $[x_{i-1}, x_i]$ (voir la Discussion 4.3.2 du Chapitre précédent).

Le choix des nœuds : Les nœuds sont choisis comme le milieu de l'intervalle d'admissibilité pour la convexité, si cet intervalle est non vide, sinon comme le milieu de l'intervalle d'admissibilité pour la monotonie.

Un exemple simple montre que l'algorithme de McAllister et Roulier est seulement d'ordre 2. En effet la fonction $f(x) = x^2$, sur l'intervalle $[0, 1]$, est approximée à l'ordre $O(h^2)$ par cet algorithme (voir le Tableau 5.1).
Ce problème est dû au fait que la moyenne harmonique h_i est seulement une approximation du premier ordre de $f'(x_i)$ quand x_i est proche d'une racine de f'. Par la suite, on présente des algorithmes qui tentent de remédier à ce problème.

5.3 Les algorithmes alternatifs

DeVore [6] a proposé de modifier la spline interpolante p quand f' est très petite afin d'améliorer la convergence. Il introduit les pentes

$$s_i = \frac{\delta_i \Delta x_{i+1} + \delta_{i+1} \Delta x_i}{\Delta x_i + \Delta x_{i+1}}, \qquad \text{pour } 1 \leq i \leq n-1, \tag{5.3}$$

où $\Delta x_i = x_i - x_{i-1}$.

DeVore a proposé de modifier les pentes de la façon suivante, pour $i = 2, \ldots, n-1$

$$d_i = \begin{cases} 0, & \text{si } \delta_i \delta_{i+1} \leq 0, \\ h_i, & \text{si } \delta_i \delta_{i+1} > 0 \text{ et } \min(\frac{s_{i-1}}{\delta_i}, \frac{s_i}{\delta_i}) \geq 2, \\ s_i, & \text{sinon}, \end{cases} \tag{5.4}$$

pour $i = 1$,

$$d_1 = \begin{cases} 0, & \text{si } \delta_1 \delta_2 \leq 0, \\ s_1, & \text{sinon}, \end{cases} \tag{5.5}$$

et aux extrémités de l'intervalle, les pentes sont données par (5.2).

Remarque 5.3.1

1. *La pente s_i donnée par (5.3) est la pente du polynôme d'interpolation du degré 2 passant par les points (x_k, y_k), $k = i-1, i, i+1$.*
2. *En choisissant le paramètre*

$$a_{Di} = \frac{\Delta x_{i+1}}{\Delta x_i + \Delta x_{i+1}},$$

on voit bien que le choix de pentes de DeVore est un cas particulier du choix de Lahtinen (voir l'équation 4.1).

Le choix des nœuds : Une fois les pentes déterminées, on choisit les nœuds ξ_i comme suit. Si l'intervalle d'admissibilité pour la convexité est non vide, on prend ξ_i comme le milieu de cet intervalle; sinon on prend ξ_i comme le milieu de l'intervalle d'admissibilité pour la monotonie, qui est non vide comme le montre le lemme suivant.

Lemme 5.3.2 *Soient les données arbitraires* $\{(x_i, y_i)\}_{i=0}^{n}$. *Pour le choix de pentes (5.4) et (5.2), les intervalles d'admissibilité pour la monotonie sont toujours non vides.*

Démonstration

. Si $\delta_i = 0$, alors $d_{i-1} = d_i = 0$ par (5.4), (5.5) ou (5.2) et par suite l'intervalle d'admissibilité pour la monotonie est tout l'intervalle $]x_{i-1}, x_i[$.

. Si $\delta_i \neq 0$, il est suffisant de montrer (3.7). On peut supposer que $d_{i-1} \neq 0$ et $d_i \neq 0$, et on considère tous les cas possibles

-*Cas* $i = 1$: l'équation (5.2) implique que δ_1 et $2\delta_1 - d_1$ sont de même signe. Supposons qu'ils sont positifs, alors $d_0 = 2\delta_1 - d_1 > 0$ et donc (3.7) est vérifiée. On obtient un résultat similaire s'ils sont négatifs.

-*Cas* $1 < i < n$: puisque $d_{i-1} \neq 0$ et $d_i \neq 0$, il s'en suit de (5.4) que δ_{i-1}, δ_i et δ_{i+1} sont de même signe, et par suite d_{i-1} et d_i le sont aussi. Supposons qu'ils sont positifs. Si d_i est donnée par h_i dans (5.4), alors $d_i = h_i = \dfrac{2\delta_i\delta_{i+1}}{\delta_i + \delta_{i+1}}$, et donc $d_i < 2\delta_i$, et par suite (3.7) est vérifiée. D'autre part, si d_i est donnée par s_i, (3.7) est obtenue par le critère même du choix de s_i dans (5.4). On obtient des résultats similaires si d_{i-1} est donnée par h_{i-1} ou s_{i-1}.

-*Cas* $i = n$: similaire au cas $i = 1$. ■

Remarque 5.3.3 *Dans le Lemme 3.3.3, la condition (3.7) nous assure que l'intervalle d'admissibilité pour la monotonie est non vide. On peut alors déterminer facilement cet intervalle grâce à la condition (3.8).*

Les deux lemmes suivants nous montrent que la spline p solution du problème d'interpolation d'Hermite, où les pentes d_i sont données par (5.4), préserve la forme des données.

Lemme 5.3.4 *Si la fonction* f *est croissante (décroissante) sur l'intervalle* $[a, b]$, *alors la spline* p *l'est aussi.*

Démonstration On sait que p préserve localement la monotonie, c'est-à-dire qu'elle est monotone sur chaque intervalle $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, car les nœuds sont choisis dans les intervalles d'admissibilité pour la monotonie. Supposons que la fonction f est croissante sur l'intervalle $[a, b]$. Dans ce cas, $y_i \geq y_{i-1}$, $i = 1, \ldots, n$, et par suite d'après (5.4) et (5.2) on a $d_i \geq 0$, $i = 0, \ldots, n$. On distingue deux cas

. Si d_{i-1} ou d_i est strictement positive, p doit être croissante sur l'intervalle $[x_{i-1}, x_i]$.

. Si $d_{i-1} = d_i = 0$, d'après (3.3)

$$\bar{d}_i = 2\delta_i - [\omega d_{i-1} + (1-\omega)d_i] = 2\delta_i \geq 0 \qquad \text{où} \quad \omega = \frac{\xi_i - x_{i-1}}{x_i - x_{i-1}},$$

et par suite, d'après (3.6) p est croissante sur l'intervalle $[x_{i-1}, x_i]$.

Ceci étant vrai pour tout $i = 1, \ldots, n$, on en déduit que la spline p est croissante sur l'intervalle $[a, b]$. On obtient des résultats similaires en supposant f décroissante sur l'intervalle $[a, b]$. ■

Lemme 5.3.5 *Si la fonction f est strictement convexe (resp. strictement concave) sur l'intervalle $[a, b]$, alors la spline p est convexe (resp. concave) sur cet intervalle.*

Démonstration Supposons que la fonction f est strictement convexe sur $[a, b]$. Dans ce cas, $\delta_i < \delta_{i+1}$, $i = 1, \ldots, n-1$.

(i) Supposons que $\delta_i \neq 0, i = 1, \ldots, n$

. Si $\delta_i \delta_{i+1} > 0$, alors d_i est ou bien la moyenne harmonique ou bien une combinaison convexe de δ_i et δ_{i+1}, ce qui implique que $\delta_i < d_i < \delta_{i+1}$.

. Si $\delta_i \delta_{i+1} < 0$, alors $d_i = 0$, et on a encore $\delta_i < d_i < \delta_{i+1}$.

Donc, on a

$$\delta_1 < d_1 < \delta_2 < \ldots < d_{n-1} < \delta_n, \tag{5.6}$$

or d'après (3.12), ceci implique que l'intervalle d'admissibilité pour la convexité en $[x_{i-1}, x_i]$ est non vide pour $i = 2, \ldots, n-1$. Et puisque le nœud ξ_i est choisi

dans cet intervalle, p est convexe sur $[x_{i-1}, x_i]$, $i = 2, \ldots, n-1$, c'est-à-dire que $d_{i-1} \leq \bar{d}_i \leq d_i$ sur cet intervalle. Et par suite, p est convexe sur l'intervalle $[x_1, x_{n-1}]$.

(ii) Si $\delta_i = 0$, pour un $i = 1, \ldots, n$, alors $d_{i-1} = d_i = 0$, et par suite l'intervalle d'admissibilité pour la convexité en $[x_{i-1}, x_i]$ est encore une fois non vide.

(iii) Montrons que l'intervalle d'admissibilité pour la convexité sur $[x_0, x_1]$ est non vide.

. Si $\delta_1 = 0$, ceci est évident car $d_0 = d_1 = 0$.

. Si $\delta_1 \neq 0$, on a deux possibilités :

- si $\delta_1(2\delta_1 - d_1) > 0$, alors par (5.2) on a $d_0 = 2\delta_1 - d_1 < \delta_1$, car $\delta_1 < d_1$ d'après (5.6).

- si $\delta_1(2\delta_1 - d_1) \leq 0$, alors par (5.2) on a $d_0 = 0$, et dans ce cas $\delta_1 > 0$, car dans le cas contraire on aura $\delta_1 \leq 0$ et $2\delta_1 - d_1 \geq 0$, c'est-à-dire $d_1 \leq 2\delta_1 \leq \delta_1$, ce qui contredit (5.6).

Donc dans les deux cas, on a $d_0 < \delta_1 < d_1$, et par suite d'après (3.12) p est convexe sur l'intervalle $[x_0, x_1]$.

(iv) De la même façon, on démontre que l'intervalle d'admissibilité pour la convexité sur $[x_{n-1}, x_n]$ est non vide. Et par suite, p est convexe sur l'intervalle $[a, b]$.

On obtient des résultats similaires pour la concavité de f sur l'intervalle $[a, b]$. ■

Remarque 5.3.6 *Dans le Lemme 5.3.5, on obtient la convexité de p en supposant que f est strictement convexe (c'est-à-dire que $\{\delta_i\}_{i=1}^n$ est strictement croissante). On peut affaiblir cette condition en permettant à $\{\delta_i\}_{i=1}^n$ d'être juste croissante. Dans ce cas on permet au nœud ξ_i d'être choisit en x_{i-1} ou x_i. Cependant, ceci implique une perte d'un degré dans l'ordre de convergence de p en ces points.*

5.4 L'ordre de convergence

Dans cette partie, on montre que l'algorithme décrit précédemment approxime à l'ordre 3 toute fonction $f \in \mathcal{C}^1[a,b]$ monotone. Sans perte de généralité, on suppose que f est croissante.

Notations On note par

- $||\,.\,|| = ||\,.\,||_{[a,b]}$ la norme du suprémum essentiel sur l'intervalle $[a,b]$:

$$||f|| = \text{ess sup}\{|f(x)|; x \in [a,b]\},$$

- $M = ||f^{(3)}||$,
- $h = \max\{x_i - x_{i-1},\ i = 1, \ldots, n\}$.

On rappelle la notion des différences divisées d'une fonction f par rapport à des points $(x_i)_{i=0}^n$: $x_0 \leq x_1 \leq x_2 \leq \ldots \leq x_n$

- $f[x_i] = f(x_i)$,

$$\bullet\ f[x_i, x_{i+1}] = \begin{cases} \dfrac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}, & \text{si } x_i \neq x_{i+1}, \\ f'(x_i), & \text{si } x_i = x_{i+1}, \end{cases}$$

$$\bullet\ f[x_i, x_{i+1}, \ldots, x_{i+k}] = \begin{cases} \dfrac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}, & \text{si } x_i < x_{i+k}, \\ \dfrac{f^{(k)}(x_i)}{k!}, & \text{si } x_i = x_{i+k}. \end{cases}$$

Et, on rappelle le théorème suivant qui va souvent être utilisé par la suite.

Théorème 5.4.1 *Supposons que $f \in \mathcal{C}^n[a,b]$, et $x_0, x_1, \ldots, x_n$ sont des réels distincts ou non dans $[a,b]$. Alors il existe $\xi \in]a,b[$ tel que :*

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

En calculant la différence divisée de f, on trouve

$$|f'(x_i) - s_i| = \Delta x_i \Delta x_{i+1} |f[x_{i-1}, x_i, x_i, x_{i+1}]| \leq \frac{1}{6} M h^2, \qquad 0 < i < n. \tag{5.7}$$

On peut trouver un résultat similaire pour $f'(x_i) - h_i, i = 2, \ldots, n-1$ pourvu que $\min(s_{i-1}, s_i) \geq 2\delta_i$.

En effet, s_i et h_i sont par définition entre δ_i et δ_{i+1}, et par suite en utilisant (5.7),

$$|f'(x_i) - h_i| \leq |f'(x_i) - s_i| + |s_i - h_i| \leq \frac{1}{6} M h^2 + |\delta_{i+1} - \delta_i|. \tag{5.8}$$

Afin d'estimer $|\delta_{i+1} - \delta_i|$, on calcule la différence divisée de f par rapport aux points x_j, $j = i-2, \ldots, i+1$, ainsi

$$f[x_{i-2}, x_{i-1}, x_i, x_{i+1}] = \frac{\dfrac{\delta_{i+1} - \delta_i}{x_{i+1} - x_{i-1}} - \dfrac{\delta_i - \delta_{i-1}}{x_i - x_{i-2}}}{x_{i+1} - x_{i-2}}. \tag{5.9}$$

On suppose que $\min(s_{i-1}, s_i) \geq 2\delta_i$. Comme s_i est une combinaison convexe de δ_i et δ_{i+1}, $s_i \geq 2\delta_i$ implique que $\delta_{i+1} \geq 2\delta_i$.

De façon similaire, on a $\delta_{i-1} \geq 2\delta_i$. Donc le numérateur dans (5.9) est la somme de deux termes positifs, et par suite d'après le Théorème 5.4.1, on a

$$f[x_{i-2}, x_{i-1}, x_i, x_{i+1}] = \frac{\dfrac{\delta_{i+1} - \delta_i}{x_{i+1} - x_{i-1}} - \dfrac{\delta_i - \delta_{i-1}}{x_i - x_{i-2}}}{x_{i+1} - x_{i-2}} = \frac{f^{(3)}(\xi)}{3!} \leq \frac{M}{6},$$

ce qui implique que

$$\frac{\delta_{i+1} - \delta_i}{x_{i+1} - x_{i-1}} \leq (x_{i+1} - x_{i-2}) \frac{M}{6},$$

et

$$|\delta_{i+1} - \delta_i| \leq \frac{3h.2h.M}{6} = Mh^2.$$

En utilisant ce dernier résultat dans (5.8), on obtient

$$|f'(x_i) - h_i| \leq 2Mh^2, \quad i = 1, \ldots, n-1, \quad \text{à condition que } \min(s_{i-1}, s_i) \geq 2\delta_i. \tag{5.10}$$

Théorème 5.4.2 *Si f est 3 fois continuement différentiable et monotone sur $[a,b]$, alors (5.4), (5.5) et (5.2) génèrent une spline quadratique p satisfaisant*

$$|| f - p || \leq 3Mh^3. \tag{5.11}$$

Démonstration Remarquons tout d'abord que

$$|f'(x_i) - d_i| \leq \begin{cases} 2Mh^2, & 0 < i < n, \\ 3Mh^2, & i = 0, n. \end{cases} \tag{5.12}$$

En effet, pour

- $0 < i < n$:

  - Si $d_i = s_i$ ou h_i, d'après (5.7) ou (5.10) respectivement.

  - Si $d_i = 0$, alors d'après (5.4) ou bien $\delta_i = 0$, ou bien $\delta_{i+1} = 0$. D'où f' s'annule sur l'intervalle contenant x_i, et (5.12) en découle.

- $i = 0$:

  - Si $d_0 = 2\delta_1 - d_1$, on a alors

$$|2\delta_1 - f'(x_0) - f'(x_1)| = (x_1 - x_0)^2 |f[x_0, x_0, x_1, x_1]| \leq \frac{1}{6} Mh^2. \tag{5.13}$$

  En utilisant (5.12) pour $i = 1$, on a

$$\begin{aligned} |2\delta_1 - d_1 - f'(x_0)| &\leq |2\delta_1 - f'(x_1) - f'(x_0)| + |f'(x_1) - d_1| \\ &\leq \frac{1}{6} Mh^2 + 2Mh^2 \\ &\leq 3Mh^2 \end{aligned}$$

  - Si $d_0 = 0$, alors d'après (5.2) on a deux possibilités :

    . si $\delta_1 \leq 0$, alors $\delta_1 = 0$, et par suite $f'(x_0) = 0$ et (5.12) est évidente.

    . si $2\delta_1 - d_1 \leq 0$, alors en utilisant (5.13) et la première partie de (5.12),

on obtient

$$\begin{aligned} f'(x_0) &= (f'(x_0) + f'(x_1) - 2\delta_1) + (2\delta_1 - d_1) + (d_1 - f'(x_1)) \\ f'(x_0) &\leq (f'(x_0) + f'(x_1) - 2\delta_1) + (d_1 - f'(x_1)) \\ &\leq \frac{1}{6}Mh^2 + 2Mh^2 \\ &\leq 3Mh^2 \end{aligned}$$

- $i = n$: On obtient le même résultat de façon similaire.

Pour toute fonction continuement différentiable g, on définit

$$d'(g) := 2\delta_i(g) - \omega g'(x_{i-1}) - (1-\omega)g'(x_i),$$

où $\delta_i(g) = \dfrac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}$, $\omega = \dfrac{\xi_i - x_{i-1}}{x_i - x_{i-1}}$ et $1 - \omega = \dfrac{x_i - \xi_i}{x_i - x_{i-1}}$.

Il s'en suit que $d'(Q) = Q'(\xi_i)$ pour tout polynôme du second degré Q. Si on considère Q comme étant le polynône de Taylor du second degré associé à f en ξ_i

$$\begin{aligned} f(x) &= f(\xi_i) + f'(\xi_i)(x - \xi_i) + \frac{1}{2}f''(\xi_i)(x - \xi_i)^2 + \frac{1}{6}f^{(3)}(c_x)(x - \xi_i)^3 \\ &= Q(x) + \frac{1}{6}f^{(3)}(c_x)(x - \xi_i)^3 \end{aligned}$$

alors $f'(x) = Q'(x) + \frac{1}{2}f^{(3)}(\tilde{c}_x)(x - \xi_i)^2$ et par suite $||f' - Q'|| \leq \frac{1}{2}Mh^2$. En utilisant le Théorème de la valeur moyenne, on a

$$|\delta_i(f - Q)| = |f'(\xi) - Q'(\xi)| \leq \frac{1}{2}Mh^2, \quad \text{avec} \quad \xi \in]x_{i-1}, x_i[.$$

Comme $f'(\xi_i) = Q'(\xi_i) = d'(Q)$, on a

$$\begin{aligned} |f'(\xi_i) - d'(f)| &= |d'(f - Q)| \\ &= |2\delta_i(f - Q) - \omega(f - Q)'(x_{i-1}) - (1 - \omega)(f - Q)'(x_i)| \\ &\leq Mh^2 + \frac{1}{2}Mh^2 \\ &\leq 2Mh^2. \end{aligned}$$

En utilisant cette dernière inégalité avec (5.12), et le fait que

$$p'(\xi_i) \;=\; \bar{d}_i \;=\; 2\delta_i - \omega d_{i-1} - (1-\omega)d_i,$$

on trouve

$$\begin{aligned} |f'(\xi_i) - p'(\xi_i)| &\leq |f'(\xi_i) - d'(f)| + |d'(f) - \bar{d}_i| \\ &\leq 2Mh^2 + \omega|f'(x_{i-1}) - d_{i-1}| + (1-\omega)|f'(x_i) - d_i| \\ &\leq 5Mh^2. \end{aligned} \tag{5.14}$$

Ceci montre que la fonction p', linéaire par morceaux, approxime f' avec une erreur inférieure ou égale à $5Mh^2$ en chaque nœud. Donc si T est une fonction linéaire par morceaux qui interpole f' en ces nœuds, alors $||T-p'|| \leq 5Mh^2$. En utilisant l'interpolation linéaire par morceaux, on a $||f' - T|| \leq \frac{1}{8}Mh^2$. Ceci implique que $||f' - p'|| \leq 6Mh^2$. Puisque $f - p$ s'annule en chaque x_i, $i = 0, \ldots, n$, on obtient (5.11) en intégrant au voisinage de x_i par rapport à x. ■

5.5 Un second algorithme

Dans la section précédente, on a présenté un algorithme qui, pour des fonctions monotones, converge à l'ordre 3. Cependant quand la fonction f change de monotonie sur l'intervalle $[a, b]$, ces mêmes arguments montrent qu'il y a convergence à l'ordre 3 seulement sur des intervalles contenus strictement dans les intervalles de monotonie. Sur les intervalles où la monotonie change, on aura généralement seulement une convergence à l'ordre 2. Ceci est dû au fait que si $\delta_i = 0$, alors p est identiquement constante sur $[x_{i-1}, x_i]$ tandis qu'en général le mieux qu'on puisse dire sur l'erreur est que

$$||f - p|| \leq C||f''||h^2,$$

où C est une constante. Le premier algorithme de DeVore attribue la valeur nulle à la pente d_i au point x_i si les données changent de monotonie, c'est-à-dire $\delta_i\delta_{i+1} \leq 0$. D'autre part, si les données proviennent d'une fonction f, tout ce qu'on sait c'est que f change de monotonie sur l'intervalle $[x_{i-1}, x_{i+1}]$. C'est la raison pour laquelle l'algorithme ne sera que d'ordre 2 au voisinage des points où la fonction change de monotonie. D'un autre côté, on n'est pas obligé d'exiger que p' change de signe exactement en x_i. Par conséquent, DeVore a proposé un second algorithme où pour $i = 2, \ldots, n-1$, (5.4) est remplacée par

$$d_i = \begin{cases} 0, & \text{si } \delta_i = 0 \text{ et } \delta_{i-1}\delta_{i+1} \geq 0 \quad \text{ou} \quad \delta_{i+1} = 0 \text{ et } \delta_i\delta_{i+2} \geq 0, \\ h_i, & \text{si } \delta_i\delta_{i+1} > 0 \quad \text{et} \quad \min(\frac{s_{i-1}}{\delta_i}, \frac{s_i}{\delta_i}) \geq 2, \\ s_i, & \text{sinon,} \end{cases} \tag{5.15}$$

Ainsi, par rapport aux pentes du premier algorithme, d_i a changé uniquement dans le cas où $\delta_i\delta_{i+1} \leq 0$.

La pente d_1 est donnée par (5.5), et les pentes aux extrémités de l'intervalle $[a, b]$ sont données par

$$\begin{aligned} d_0 &= 2\delta_1 - d_1, \\ d_n &= 2\delta_n - d_{n-1}. \end{aligned} \tag{5.16}$$

Le choix des nœuds : Les nœuds ξ_i sont définis de la façon suivante. Si l'intervalle d'admissibilité pour la convexité est non vide, on prend ξ_i comme le milieu de cet intervalle; sinon on prend ξ_i comme le milieu de l'intervalle d'admissibilité pour la monotonie s'il est non vide; dans le cas contraire on prend ξ_i comme le milieu de l'intervalle $]x_{i-1}, x_i[$.

Théorème 5.5.1 *Si f est 3 fois continuement différentiable sur $[a, b]$, alors le second algorithme de DeVore génère une spline quadratique p satisfaisant*

$$||f - p|| \leq 3Mh^3. \tag{5.17}$$

Démonstration Tout d'abord on montre que

$$|f'(x_i) - d_i| \leq 2Mh^2, \qquad i = 1, \ldots, n-1. \tag{5.18}$$

On considère les trois possibilités pour le choix de pentes d_i données par (5.15)

. Si $d_i = s_i$, alors (5.7) est vérifiée pour n'importe quelle $f \in \mathcal{C}^3([a,b])$, d'où (5.18).

. Si $d_i = h_i$, alors $\min(\frac{s_{i-1}}{\delta_i}, \frac{s_i}{\delta_i}) \geq 2$ et $\delta_i \delta_{i+1} > 0$, et par suite (5.10) est vérifiée, d'où (5.18).

. Si d_i est donnée par la première ligne de (5.15), alors on a deux cas :

- Si $\delta_i = 0$ et $\delta_{i-1}\delta_{i+1} \geq 0$, on calcule la différence divisée de f aux points $x_{i-1}, x_i, x_{i+1}, x_{i+2}$ comme dans (5.9). Puisque le numérateur dans (5.9) est la somme de deux termes de même signe, on obtient $|\delta_{i+1} - \delta_i| \leq Mh^2$. On a $\delta_i = 0$ et s_i est entre δ_i et δ_{i+1}, d'où d'après (5.7) on trouve

$$\begin{aligned} |f'(x_i) - 0| &\leq |f'(x_i) - s_i| + |s_i - \delta_i| \\ &\leq |f'(x_i) - s_i| + |\delta_{i+1} - \delta_i| \\ &\leq Mh^2 + Mh^2 = 2Mh^2, \end{aligned}$$

d'où (5.18).

- Si $\delta_{i+1} = 0$ et $\delta_i \delta_{i+2} \geq 0$, on obtient le même résultat de façon similaire en calculant la différence divisée de f aux points $x_i, x_{i+1}, x_{i+2}, x_{i+3}$.

Ainsi (5.18) est démontrée.

D'après (5.13), on a $|f'(x_0) - d_0| \leq 3Mh^2$. La même inégalité est toujours vraie pour $i = n$, d'où

$$|f'(x_i) - d_i| \leq 3Mh^2, \qquad i = 0 \text{ et } i = n. \tag{5.19}$$

À partir de (5.18) et (5.19), on peut compléter la preuve exactement comme pour le Théorème 5.4.2. ■

Théorème 5.5.2 *La spline p générée par le deuxième algorithme de DeVore change de monotonie sur l'intervalle $[x_1, x_{n-1}]$ autant de fois que la suite $\delta : \delta_1, \ldots, \delta_n$ change de signe.*

Démonstration On va montrer que p' change de signe sur $[x_1, x_{n-1}]$ autant de fois que δ. À partir de la définition de d_i, $i = 1, \ldots, n-1$, on voit que d_i est positif quand δ_i et δ_{i+1} sont tous les deux positifs et d_i est négatif si δ_i et δ_{i+1} sont tous les deux négatifs. Donc la suite $\lambda : \delta_1, d_1, \delta_2, d_2, \ldots, \delta_{n-1}, d_{n-1}, \delta_n$ change de signe autant de fois que δ. On veut insérer $\bar{d}_i = p'(\xi_i)$ dans la suite λ entre d_{i-1} et d_i, $i = 2, \ldots, n-1$ de sorte que la suite résultante λ^* change de signe autant de fois que λ. Le seul cas non trivial est quand d_{i-1}, δ_i et d_i sont tous de même signe (ou nuls). On va montrer que dans ce cas, l'intervalle d'admissibilité pour la monotonie sur $[x_{i-1}, x_i]$ est non vide, c'est-à-dire que la condition (3.7) du Lemme 3.3.3 est vérifiée (voir la Remarque 5.3.3). Ainsi, on peut très bien inclure $\bar{d}_i$ dans ce cas.
On suppose que d_{i-1}, δ_i et d_i sont tous positifs, le cas où ils sont négatifs est traité de la même façon. On considère tous les cas possibles

. Cas $\delta_i > 0$:

- $d_{i-1} = 0$ ou $d_i = 0$, alors (3.7) est immédiate.
- $d_{i-1} = h_{i-1}$ ou $d_i = h_i$, alors soit $d_{i-1} < 2\delta_i$ soit $d_i < 2\delta_i$. Dans les deux cas (3.7) est vérifiée.
- $d_{i-1} = s_{i-1}$ et $d_i = s_i$, alors ou bien $\delta_{i+1} > 0$ et dans ce cas par le critère de sélection de (5.15) on a (3.7), ou bien $\delta_{i+1} \leq 0$ et donc $d_i < \delta_i$ (car d_i est une combinaison convexe de δ_i et δ_{i+1}) et par suite (3.7) est vérifiée.

. Cas $\delta_i = 0$:

- $d_{i-1} = d_i = 0$, dans ce cas l'intervalle d'admissibilité pour la monotonie est $]x_{i-1}, x_i[$.
- $d_{i-1} > 0$ et $d_i = 0$, alors on doit avoir $d_{i-1} = s_{i-1}$ d'après (5.15), et comme s_{i-1} est entre δ_{i-1} et δ_i on en déduit que $\delta_{i-1} > 0$. Le premier critère de choix de (5.15) implique que $\delta_{i+1} < 0$ (sinon d_{i-1} est nulle). Mais on doit aussi avoir $d_i = s_i$ d'après (5.15) car les deux autres critères ne sont pas satisfaits. Et

puisque s_i est une combinaison convexe de δ_i et δ_{i+1} on en déduit que $s_i < 0$, ce qui contredit l'hypothèse et nous prouve qu'on ne peut pas rencontrer ce cas.

- $d_{i-1} = 0$ et $d_i > 0$, similaire au cas précédent.

Ainsi on a montré qu'il est possible de créer une suite λ^*, comme décrite précédemment, qui change de signe autant de fois que λ. Maintenant considérons la suite τ : $d_1, \bar{d}_2, d_2, \bar{d}_3, \ldots, d_{n-1}$ qui est une sous-suite de λ^* et qui change de signe autant de fois que λ^* (donc autant de fois que δ). Et puisque p' change de signe sur $[x_1, x_{n-1}]$ autant de fois que τ, on en déduit le résultat. ■

5.6 Des exemples numériques

Dans cette section, on donne des exemples de fonctions régulières et on analyse l'erreur d'approximation de l'algorithme de McAllister et Roulier et des deux algorithmes présentés par DeVore. Dans ces exemples, on prend une subdivision de $n+1$ points également espacés de l'intervalle $[0, 1]$ et on estime l'erreur en utilisant des algorithmes qui déterminent le maximum d'une fonction (voir le Tableau 5.1).

5.7 Conclusion

Le premier algorithme de DeVore est d'ordre $O(h^3)$ pour toute fonction f monotone sur l'intervalle $[a, b]$. Cependant, il est en général seulement d'ordre $O(h^2)$ au voisinage des points de changement de monotonie.

Le deuxième algorithme de DeVore est peut-être plus intéressant, dans le sens où il est d'ordre $O(h^3)$ pour n'importe quelle fonction $f \in \mathcal{C}^3([a, b])$. Cet algorithme préserve la

Tableau 5.1 – *Exemple d'analyse d'erreur donné par* `DeVore`

	n	McAllister et Roulier	Premier algorithme	Deuxième algorithme
$y = x^2$	16	0.488281250000d-03	0.138777878078d-16	résultats
	32	0.122070312500d-03	0.277555756156d-16	identiques à
	64	0.305175781250d-04	0.138777878078d-16	ceux du premier
	128	0.762939453125d-05	0.277555756156d-16	algorithme
	256	0.190734864281d-05	0.277555756156d-16	de DeVore
		$O(h^2)$	zéro	
$y = cos(x)$	16	0.277829405296d-03	0.126783470478d-04	résultats
	32	0.560392383724d-04	0.161480136285d-05	identiques à
	64	0.157472067168d-04	0.203664441756d-06	ceux du premier
	128	0.387501773202d-05	0.255695074003d-07	algorithme
	256	0.961169510830d-06	0.320309312407d-08	de DeVore
		$O(h^2)$	$O(h^3)$	
$y = xsin(x)$	32	0.130084133320d-03	0.591354137214d-05	résultats
	64	0.314947442995d-04	0.743824330129d-06	identiques à
	128	0.775005548855d-05	0.932565455969d-07	ceux de premier
	256	0.192234253500d-05	0.116741301071d-07	algorithme
	512	0.478705420260d-06	0.146032175241d-08	de DeVore
		$O(h^2)$	$O(h^3)$	
$y = cos(6x)$	32	0.277455600555d-02	0.371189149842d-02	0.294413496052d-03
	64	0.567109080345d-03	0.104923798966d-02	0.363051687600d-04
	128	0.277341531626d-03	0.276519887848d-03	0.448985110779d-05
	256	0.667778913560d-04	0.655773692415d-04	0.802927047516d-06
	512	0.152910747667d-04	0.143150564327d-04	0.979241505661d-07
		$O(h^2)$	$O(h^2)$	$O(h^3)$

monotonie mais dans un sens qu'il faudrait préciser :

si f est monotone sur l'intervalle $I = [x_{i-1}, x_i]$, alors la spline interpolante p est monotone sur un plus petit intervalle contenu dans I.

Finalement, ces deux algorithmes préservent la convexité des données.

CONCLUSION

1. Les méthodes présentées dans ce mémoire ont l'avantage d'être des méthodes *locales* (voir la Figure 2.8). Ceci est dû au caractère *local* des splines quadratiques interpolantes. Grâce à cette propriété, on peut facilement

   (i) ajuster une valeur donnée : si une spline p a été évaluée pour un ensemble de données $\{(x_i, y_i)\}_{i=0}^{n}$, et que l'on désire changer la i ème donnée (en changeant y_i et/ou en déplaçant x_i à l'intérieur de l'intervalle $[x_{i-1}, x_{i+1}]$), alors il n'est pas nécessaire de recalculer tous les coefficients d'une nouvelle spline. En effet, il suffit de considérer le problème d'interpolation sur l'intervalle $[x_{i-1}, x_{i+1}]$, et d'utiliser ensuite les nouveaux nœuds et coefficients évalués sur cet intervalle, avec ceux déjà calculés.

   (ii) prolonger les données : Il y a certaines applications où les données nous proviennent de façon continue (par exemple, en télémétrie). Dans ce cas, on ne peut pas attendre d'avoir toutes les données pour calculer la spline interpolante. En effet, on développe un algorithme *progressif*, dans le sens où la spline est, en premier lieu, évaluée pour les n premières données. Ensuite, au fur et à mesure que l'on obtient de nouvelles données, on évalue la spline interpolante sans recalculer les nœuds et coefficients déjà obtenus.

2. Toutes les méthodes citées dans ce mémoire présentent un phénomène *pathologique*,

dans le sens où la spline interpolante risque de ne pas respecter la convexité des données. Cette situation a lieu lorsque deux pentes adjacentes sont non nulles et de même valeur, c'est-à dire $\delta_i = \delta_{i+1} \neq 0$ (voir les Figures 4.2 et 4.3). Pour remédier à ce problème, on peut adopter l'une des approches suivantes

(a) la première approche consiste à assigner aux points x_{i-1} et x_{i+1} la pente d_i. Ainsi, on oblige l'algorithme à générer une spline, qui est linéaire sur l'intervalle $[x_{i-1}, x_{i+1}]$ (voir les Figures 6.2b et 6.2d). Dans ce cas, la convexité des données est préservée. Cependant il existe une situation *critique* dans le cas où

$$\delta_{i-1} = \delta_i \neq \delta_{i+1} = \delta_{i+2}.$$

La Figure 6.1 présente un exemple illustrant une pareille situation. Les x_i et y_i sont donnés dans le tableau suivant

Tableau 6.1 – *Exemple de situation critique*

x_i	0	1	2	3	5	7	9
y_i	0	2	4	6	7	8	9

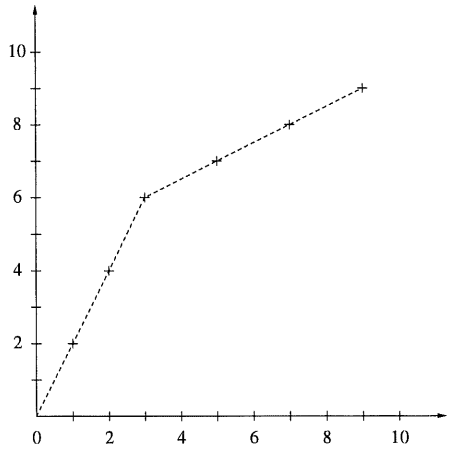


Figure 6.1 – *Exemple de situation critique*

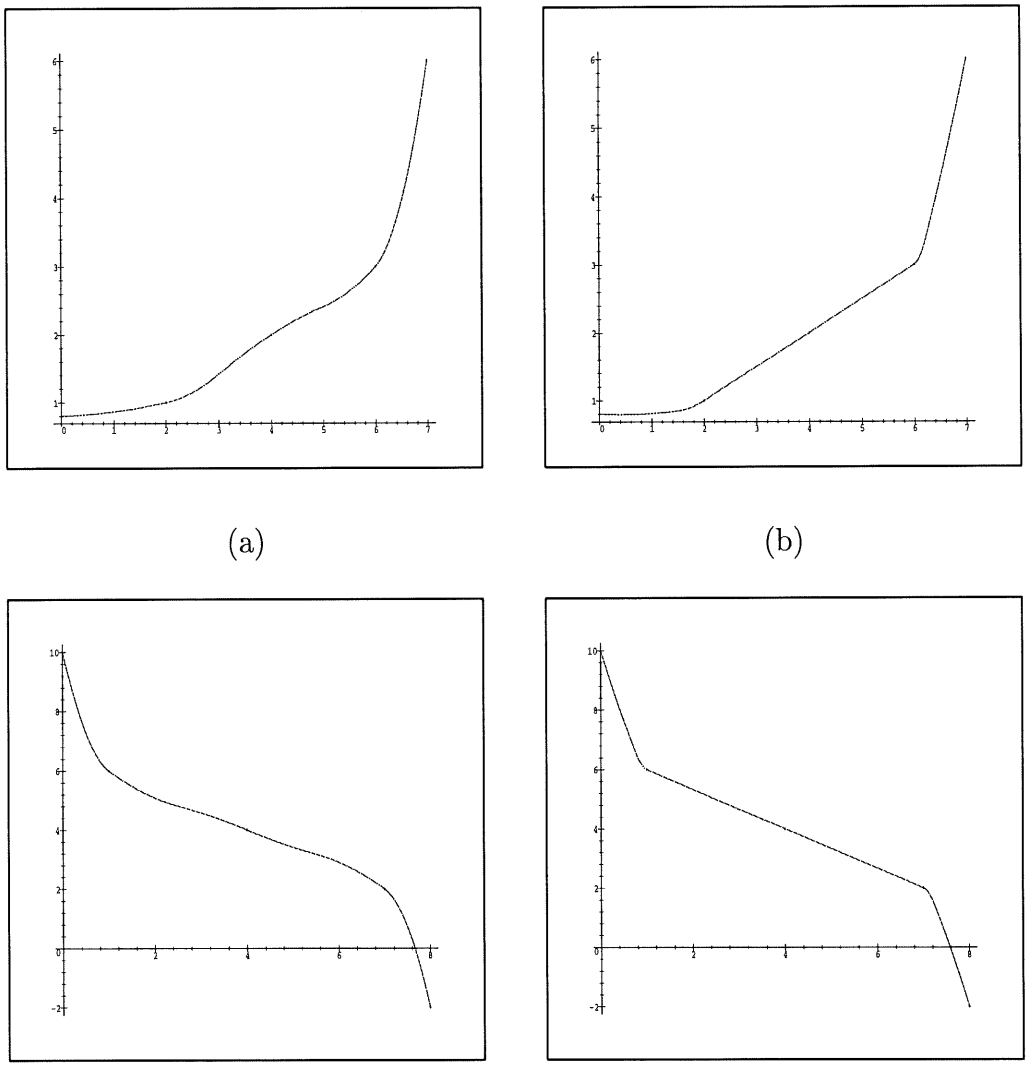
Aussi, si $\delta_i = \delta_{i+1} \neq 0$ et δ_{i-1} ou $\delta_{i+2} = 0$, on ne peut pas assigner aux points x_{i-1} et x_{i+1} la pente d_i. Dans ce cas, la convexité et la monotonie des données ne peuvent pas être préservées en même temps.

Dans ces cas particuliers, le phénomène *pathologique* persiste, et la deuxième approche s'impose.

(b) la deuxième approche consiste à enlever le coin (x_i, y_i) et à le remplacer par deux points (x_{ig}, y_{ig}) et (x_{id}, y_{id}) qui l'encadrent.

Le nouveau point (x_{ig}, y_{ig}) est choisi sur la corde reliant (x_{i-1}, y_{i-1}) et (x_i, y_i) à une distance de εL_i de ce dernier point; L_i étant la *longueur* de la corde reliant les deux points successifs (x_{i-1}, y_{i-1}) et (x_i, y_i) (voir l'équation 3.20) et ε, un paramètre tel que $0 < \varepsilon < 1$. Le point (x_{id}, y_{id}) est déterminé sur la corde reliant (x_i, y_i) et (x_{i+1}, y_{i+1}) de façon similaire.

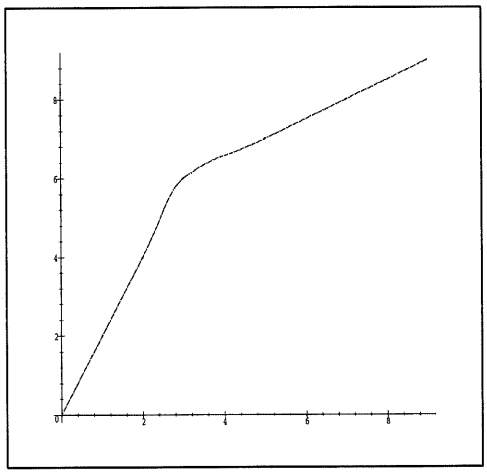
La spline est ensuite évaluée en remplaçant (x_i, y_i) par ces nouveaux points. Ainsi, on peut développer un algorithme noté *ε-algorithme*, qui génère une spline qui est d'autant plus proche du point (x_i, y_i) que ε est petit (voir la Figure 6.3). Dans ce cas, la convexité des données est préservée.



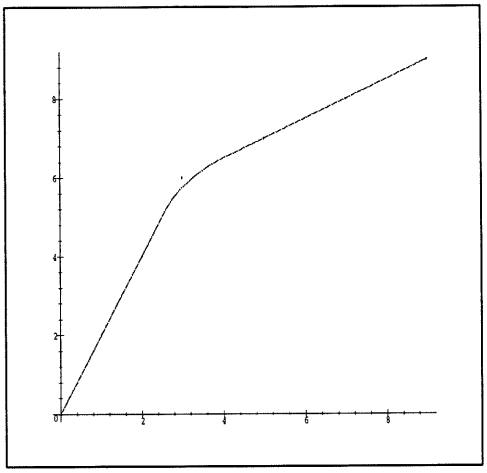
(a) (b)

(c) (d)

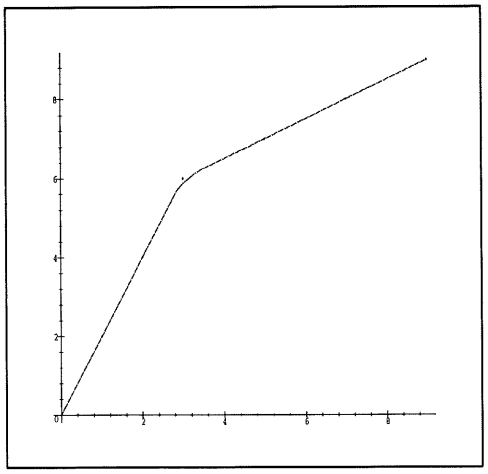
Figure 6.2 – *La première approche*



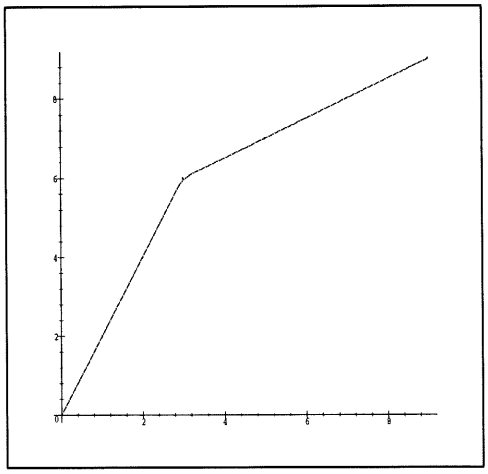
(a) algorithme initial : la convexité n'est pas préservée



(b) *ε-algorithme* avec $\varepsilon = 0.5$



(c) *ε-algorithme* avec $\varepsilon = 0.25$



(d) *ε-algorithme* avec $\varepsilon = 0.125$

Figure 6.3 – *Application de l'ε-algorithme*

ANNEXES

Dans cette appendice, on présente une codification des principaux algorithmes présentés dans le mémoire.

Le premier algorithme, celui de McAllister et Roulier [16], est codifié en Fortran. Son programme principal fait appel à des sous-programmes appelés *subroutines* :

. Subroutine Slopes : on fait appel à ce sous-programme pour calculer les pentes $\{d_i\}_{i=0}^{n}$ à partir de (5.1) et (5.2). Ce qui permet d'ajouter au plus un nœud dans chaque sous-intervalle. Si l'utilisateur désire utiliser ses propres pentes, il peut se passer de cette *subroutine*.

. Subroutine Meval : composée de quatres *subroutines*

- Subroutine Search : permet de déterminer le sous-intervalle dans lequel on désire évaluer la spline p à partir d'un argument donné.
- Subroutine Choose : permet de choisir un des quatre cas possibles déterminés par la méthode de McAllister et Roulier.
- Subroutine Cases : permet, une fois le cas déterminé, de calculer les nœuds et coefficients nécessaires au calcul de la spline.
- Subroutine Spline : permet de localiser l'argument par rapport aux nœuds dans l'intervalle, et d'évaluer le polynôme de Bernstein.

Dans le but de comparer l'algorithme de McAllister et Roulier et celui de Schumaker, on a codifié ce dernier en C++. La codification a été faite pour les deux versions simple et interactive. Cependant, seule la première version est présentée dans cette annexe.

On obtient l'algorithme d'Iqbal en remplaçant les pentes de Schumaker par celles de Butland. Cet algorithme a été programmé et appliqué sur des exemples. En plus d'être un algorithme facile à programmer (comme celui de Schumaker), l'algorithme d'Iqbal nous permet d'avoir les résultats de façon directe (voir le troisiéme point de la discussion 4.3.2).

Tous les autres algorithmes sont des modifications des algorithmes de McAllister et Roulier et de Schumaker, et peuvent en être déduits en modifiant légèrement ces deux derniers.

Les figures présentées dans ce mémoire sont le fruit d'applications de ces algorithmes.

A.1 Algorithme de McAllister et Roulier

```
C           PROGRAM MAINDR
C
            REAL X(100),Y(100),XVAL(152),YVAL(152),D(100),DELTAH,EPS
            INTEGER N,K,ERR
            CHARACTER*40 NDFE,NDFS
C
C           THIS PROGRAM IS A TEST DRIVER FOR THE SHAPE PRESERVING QUADRATIC SPLINES
C           BY D.F.MCALLISTER AND J.A.ROULIER.
C
C           ON INPUT-
C
C           X CONTAINS THE ABSCISSAS OF THE POINTS OF INTERPOLATION.
C           Y CONTAINS THE ORDINATES OF THE POINTS OF INTERPOLATION.
C           N IS THE NUMBER OF DATA POINTS.
C           K IS THE NUMBER OF POINTS AT WHICH THE SPLINE IS TO BE EVALUATED.
C
C           UPON EXIT FROM SUBROUTINE 'SLOPES'-
C
C           D CONTAINS THE COMPUTED FIRST DERIVATIVES AT EACH DATA POINT.
C
C           SUPPLY THE UNIT NUMBERS FOR I/O OPERATIONS.
C
            IN=8
            OUT=9
C
            WRITE (*,10)
     10     FORMAT ('QUEL EST LE NOM DU FICHIER D"ENTREE')
            READ (*,*) NDFE
            OPEN(IN,FILE=NDFE)
            WRITE (*,20)
     20     FORMAT ('QUEL EST LE NOM DU FICHIER DE SORTIE')
            READ (*,*) NDFS
            OPEN(OUT,FILE=NDFS)
            READ (IN,30)N
     30     FORMAT (I3)
            READ (IN,40) (X(I),Y(I),I=1,N)
     40     FORMAT (2F15.10)
            CLOSE(IN)
C
C           CALCULATE THE SLOPES AT EACH DATA POINT.
            CALL SLOPES(X,Y,D,N)
```

```
C
C           SET THE ERROR TOLERANCE EPS, WHICH IS USED IN SUBROUTINE 'CHOOSE'.
            EPS=1.E-04
C
C           EVALUATE EQUALLY SPACED POINTS IN THE ENTIRE INTERVAL DETERMINED BY
C           THE DATA.
C
C           COMPUTE THE POINTS OF EVALUATION.
            K=150
            XVAL(1)=X(1)
            DELTAH=(X(N)-X(1))/FLOAT(K-1)
            DO 50 I=2,K
               XVAL(I)=X(1)+FLOAT(I-1)*DELTAH
   50       CONTINUE
C
            CALL MEVAL(XVAL,YVAL,X,Y,D,N,K,EPS,ERR)
            WRITE(OUT,60) (XVAL(I),YVAL(I),I=1,K)
   60       FORMAT(2F15.10)
            CLOSE(OUT)
C
            STOP
            END
```

```
      SUBROUTINE SLOPES(XTAB,YTAB,DTAB,NUM)
C
      REAL XTAB(NUM),YTAB(NUM),DTAB(NUM),D1,D2,XBAR,XHAT,YDIF1,YDIF2,YXMID,XMID,
     *     D1S,D2S
C
C     SLOPES CALCULATES THE DERIVATIVE AT EACH OF THE DATA POINTS. THE SLOPES
C     PROVIDED WILL INSURE THAT AN OSCULATORY QUADRATIC SPLINE WILL HAVE ONE
C     ADDITIONAL KNOT BETWEEN TWO ADJACENT POINTS OF INTERPOLATION.
C     CONVEXITY AND MONOTONICITY ARE PRESERVED WHEREVER THESE CONDITIONS
C     ARE COMPATIBLE WITH THE DATA.
C
C     ON INPUT-
C
C     XTAB CONTAINS THE ABSCISSAS OF THE DATA POINTS.
C     YTAB CONTAINS THE ORDINATES OF THE DATA POINTS.
C     NUM IS THE NUMBER OF DATA POINTS.
C
C     ON OUTPUT-
C
C     DTAB CONTAINS THE VALUE OF THE FIRST DERIVATIVE AT EACH DATA POINT.
C
C     AND
C
C     SLOPES DOES NOT ALTER XTAB,YTAB,NUM.
C
      NUM1=NUM-1
      IM1=1
      I=2
      I1=3
C
C     CALCULATE THE SLOPES OF THE TWO LINES JOINING THE FIRST THREE DATA POINTS.
      YDIF1=YTAB(2)-YTAB(1)
      YDIF2=YTAB(3)-YTAB(2)
      D1=YDIF1/(XTAB(2)-XTAB(1))
      D1S=D1
      D2=YDIF2/(XTAB(3)-XTAB(2))
      D2S=D2
C
C     IF ONE OF THE PRECEDING SLOPES IS ZERO OR IF THEY HAVE OPPOSITE SIGN,
C     ASSIGN THE VALUE ZERO TO THE DERIVATIVE AT THE MIDDLE POINT.
   10 IF(D1.EQ.0.E0.OR.D2.EQ.0.E0.OR.(D1*D2).LE.0.E0) GO TO 20
      IF(ABS(D1).GT.ABS(D2)) GO TO 30
      GO TO 40
```

```
   20       DTAB(I)=0.E0
            GO TO 50
C
C           CALCULATE THE SLOPE BY EXTENDING THE LINE WITH SLOPE D1.
   30       XBAR=(YDIF2/D1)+XTAB(I)
            XHAT=(XBAR+XTAB(I1))/2.E0
            DTAB(I)=YDIF2/(XHAT-XTAB(I))
            GO TO 50
C
C           CALCULATE THE SLOPE BY EXTENDING THE LINE WITH SLOPE D2.
   40       XBAR=(-YDIF1/D2)+XTAB(I)
            XHAT=(XTAB(IM1)+XBAR)/2.E0
            DTAB(I)=YDIF1/(XTAB(I)-XHAT)
C
C           INCREMENT COUNTERS.
   50       IM1=I
            I=I1
            I1=I1 + 1
            IF(I.GT.NUM1) GO TO 60
C
C           CALCULATE THE SLOPES OF THE TWO LINES JOINING THREE CONSECUTIVE DATA POINTS.
            YDIF1=YTAB(I)-YTAB(IM1)
            YDIF2=YTAB(I1)-YTAB(I)
            D1=YDIF1/(XTAB(I)-XTAB(IM1))
            D2=YDIF2/(XTAB(I1)-XTAB(I))
            GO TO 10
C
C           CALCULATE THE SLOPE AT THE LAST POINT, XTAB(NUM).
   60       IF((D1*D2).LT.0.E0) GO TO 80
            XMID=(XTAB(NUM1)+XTAB(NUM))/2.E0
            YXMID=DTAB(NUM1)*(XMID-XTAB(NUM1))+YTAB(NUM1)
            DTAB(NUM)=(YTAB(NUM)-YXMID)/(XTAB(NUM)-XMID)
            IF((DTAB(NUM)*D2).LT.0.E0) GO TO 70
            GO TO 90
   70       DTAB(NUM)=0.E0
            GO TO 90
   80       DTAB(NUM)=2.E0*D2
C
C           CALCULATE THE SLOPE AT THE FIRST POINT, XTAB(1).
   90       IF((D1S*D2S).LT.0.E0) GO TO 110
            XMID=(XTAB(1)+XTAB(2))/2.E0
            YXMID=DTAB(2)*(XMID-XTAB(2))+YTAB(2)
            DTAB(1)=(YXMID-YTAB(1))/(XMID-XTAB(1))
```

```
            IF((DTAB(1)*D1S).LT.0.E0) GO TO 100
            RETURN
            DTAB(1)=0.E0
            RETURN
  110       DTAB(1)=2.E0*D1S
            RETURN
C
            END
```

```
      SUBROUTINE MEVAL(XVAL,YVAL,XTAB,YTAB,DTAB,NUM,NUME,EPS,ERR)
C
      REAL XVAL(NUME),YVAL(NUME),XTAB(NUM),YTAB(NUM),DTAB(NUM),V1,V2,W1,W2,Z1,Z2,
     *     Y1,Y2,E1,E2,SPLINE,EPS
      INTEGER START,START1,END,END1,ERR,FND
C
C     MEVAL CONTROLS THE EVALUATION OF AN OSCULATORY QUADRATIC SPLINE. THE USER
C     MAY PROVIDE HIS OWN SLOPES AT THE POINTS OF INTERPOLATION OR USE THE SUB-
C     ROUTINE 'SLOPES' TO CALCULATE SLOPES WHICH ARE CONSISTENT WITH THE SHAPE
C     OF THE DATA.
C
C     ON INPUT-
C
C     XVAL MUST BE A NONDECREASING VECTOR OF POINTS AT WHICH THE SPLINE WILL
C     BE EVALUATED.
C     XTAB CONTAINS THE ABSCISSAS OF THE DATA POINTS TO BE INTERPOLATED.
C     XTAB MUST BE INCREASING.
C     YTAB CONTAINS THE ORDINATES OF THE DATA POINTS TO BE INTERPOLATED.
C     DTAB CONTAINS THE SLOPE OF THE SPLINE AT EACH POINT OF INTERPOLATION.
C     NUM IS THE NUMBER OF DATA POINTS (DIMENSION OF XTAB AND YTAB).
C     NUME IS THE NUMBER OF POINTS OF EVALUATION (DIMENSION OF XVAL AND YVAL).
C     EPS IS A RELATIVE ERROR TOLERANCE USED IN SUBROUTINE 'CHOOSE' TO DISTINGUISH
C     THE SITUATION DTAB(I) OR DTAB(I+1) IS RELATIVELY CLOSE TO THE SLOPE OR
C     TWICE THE SLOPE OF THE LINEAR SEGMENT BETWEEN XTAB(I) AND XTAB(I+1). IF THIS
C     SITUATION OCCURS, ROUNDOFF MAY CAUSE A CHANGE IN CONVEXITY OR MONOTONICITY
C     OF THE RESULTING SPLINE AND A CHANGE IN THE CASE NUMBER PROVIDED BYCHOOSE.
C     IF EPS IS NOT EQUAL TO ZERO, THEN EPS SHOULD BE GREATER THAN OR EQUAL TO
C     MACHINE EPSILON.
C
C     ON OUTPUT-
C
C     YVAL CONTAINS THE IMAGES OF THE POINTS IN XVAL.
C
C     ERR IS AN ERROR CODE-
C        ERR=0 - MEVAL RAN NORMALLY.
C        ERR=1 - XVAL(I) IS LESS THAN XTAB(1) FOR AT LEAST ONE I OR XVAL(I) IS GREATER
C                THAN XTAB(NUM) FOR AT LEAST ONE I. MEVAL WILL EXTRAPOLATE TO PROVIDE
C                FUNCTION VALUES FOR THESE ABSCISSAS.
C        ERR=2 - XVAL(I+1).LT.XVAL(I) FOR SOME I.
C
C     AND
C
C     MEVAL DOES NOT ALTER XVAL,XTAB,YTAB,DTAB,NUM,NUME.
```

```
C            MEVAL CALLS THE FOLLOWING SUBROUTINES OR FUNCTIONS:
C              SEARCH
C              CASES
C              CHOOSE
C              SPLINE
C
               START=1
               END=NUME
               ERR=0
               IF(NUME.EQ.1) GO TO 20
C
C              DETERMINE IF XVAL IS NONDECREASING.
               NUME1=NUME-1
               DO 10 I=1,NUME1
                  IF(XVAL(I+1).GE.XVAL(I)) GO TO 10
                  ERR=2
                  GO TO 230
    10         CONTINUE
C
C              IF XVAL(I).LT.XTAB(1), THEN XVAL(I)=YTAB(1).
C              IF XVAL(I).GT.XTAB(NUM), THEN XVAL(I)=YTAB(NUM).
C
C              DETERMINE IF ANY OF THE POINTS IN XVAL ARE LESS THAN THE ABSCISSA OF THE
C              FIRST DATA POINT.
    20         DO 30 I=1,NUME
                  IF(XVAL(I).GE.XTAB(1)) GO TO 40
                  START=I+1
    30         CONTINUE
C
    40         NUME1=NUME+1
C
C              DETERMINE IF ANY OF THE POINTS IN XVAL ARE GREATER THAN THE ABSCISSA OF THE
C              LAST DATA POINT.
               DO 50 I=1,NUME
                  IND=NUME1-I
                  IF(XVAL(IND).LE.XTAB(NUM)) GO TO 60
                  END=IND-1
    50         CONTINUE
C
C              CALCULATE THE IMAGES OF POINTS OF EVALUATION WHOSE ABSCISSAS ARE LESS THAN
C              THE ABSCISSA OF THE FIRST DATA POINT.
    60         IF(START.EQ.1) GO TO 80
C              SET THE ERROR PARAMETER TO INDICATE THAT EXTRAPOLATION HAS OCCURRED.
```

```
      ERR=1
      CALL CHOOSE(XTAB(1),YTAB(1),DTAB(1),DTAB(2),XTAB(2),YTAB(2),EPS,NCASE)
      CALL CASES(XTAB(1),YTAB(1),DTAB(1),DTAB(2),XTAB(2),YTAB(2),E1,E2,V1,V2,W1,
     *           W2,Z1,Z2,Y1,Y2,NCASE)
      START1=START-1
      DO 70 I=1,START1
         YVAL(I)= SPLINE(XVAL(I),Z1,Z2,XTAB(1),YTAB(1),XTAB(2),YTAB(2),Y1,Y2,E2,
     *           W2,V2,NCASE)
   70 CONTINUE
      IF(NUME.EQ.1) GO TO 230
C
C     SEARCH LOCATES THE INTERVAL IN WHICH THE FIRST IN-RANGE POINT OF EVALUATION
C     LIES.
   80 IF((NUME.EQ.1).AND.(END.NE.NUME)) GO TO 200
      CALL SEARCH(XTAB,NUM,XVAL(START),LCN,FND)
C
      LCN1=LCN+1
C
C     IF THE FIRST IN-RANGE POINT OF EVALUATION IS EQUAL TO ONE OF THE DATA POINTS,
C     ASSIGN THE APPROPRIATE VALUE FROM YTAB. CONTINUE UNTIL A POINT OF EVALUA-
C     TION IS FOUND WHICH IS NOT EQUAL TO A DATA POINT.
      IF(FND.EQ.0) GO TO 130
   90 YVAL(START)=YTAB(LCN)
      START1=START
      START=START+1
      IF(START.GT.NUME) GO TO 230
      IF(XVAL(START1).EQ.XVAL(START)) GO TO 90
C
  100 IF(XVAL(START)-XTAB(LCN1)) 130, 110, 120
C
  110 YVAL(START)=YTAB(LCN1)
      START1=START
      START=START+1
      IF(START.GT.NUME) GO TO 230
      IF(XVAL(START).NE.XVAL(START1)) GO TO 120
      GO TO 110
C
  120 LCN=LCN1
      LCN1=LCN1+1
      GO TO 100
C
C     CALCULATE THE IMAGES OF ALL THE POINTS WHICH LIE WITHIN RANGE OF THE DATA.
C
```

```
  130     IF((LCN.EQ.1).AND.(ERR.EQ.1)) GO TO 140
          CALL CHOOSE(XTAB(LCN),YTAB(LCN),DTAB(LCN),DTAB(LCN1),XTAB(LCN1),YTAB(LCN1),
          *           EPS,NCASE)
          CALL CASES(XTAB(LCN),YTAB(LCN),DTAB(LCN),DTAB(LCN1),XTAB(LCN1),YTAB(LCN1),
          *           E1,E2,V1,V2,W1,W2,Z1,Z2,Y1,Y2,NCASE)
C
  140     DO 190 I=START,END
C
C         IF XVAL(I)-XTAB(LCN1) IS NEGATIVE, DO NOT RECALCULATE THE PARAMETERS FOR
C         THIS SECTION OF THE SPLINE SINCE THEY ARE ALREADY KNOWN.
          IF(XVAL(I)-XTAB(LCN1)) 150,160,170
C
  150     YVAL(I)=SPLINE(XVAL(I),Z1,Z2,XTAB(LCN),YTAB(LCN),
          *           XTAB(LCN1),YTAB(LCN1),Y1,Y2,E2,W2,V2,NCASE)
          GO TO 190
C
C         IF XVAL(I) IS A DATA POINT, ITS IMAGE IS KNOWN.
  160     YVAL(I)=YTAB(LCN1)
          GO TO 190
C
C         INCREMENT THE POINTERS WHICH GIVE THE LOCATION IN THE DATA VECTOR.
  170     LCN=LCN1
          LCN1=LCN+1
C
C         DETERMINE THAT THE ROUTINE IS IN THE CORRECT PART OF THE SPLINE.
          IF(XVAL(I)-XTAB(LCN1)) 180,160,170
C
C         CALL CHOOSE TO DETERMINE THE APPROPRIATE CASE AND THEN CALL CASES TO COM-
C         PUTE THE PARAMETERS OF THE SPLINE.
  180     CALL CHOOSE(XTAB(LCN),YTAB(LCN),DTAB(LCN),DTAB(LCN1),XTAB(LCN1),YTAB(LCN1),
          *           EPS,NCASE)
          CALL CASES(XTAB(LCN),YTAB(LCN),DTAB(LCN),DTAB(LCN1),XTAB(LCN1),YTAB(LCN1),
          *           E1,E2,V1,V2,W1,W2,Z1,Z2,Y1,Y2,NCASE)
          GO TO 150
  190     CONTINUE
C
C         CALCULATE THE IMAGES OF THE POINTS OF EVALUATION WHOSE ABSCISSAS ARE GREA-
C         TER THAN THE ABSCISSA OF THE LAST DATA POINT.
          IF(END.EQ.NUME) GO TO 230
          IF((LCN1.EQ.NUM).AND.(XVAL(END).NE.XTAB(NUM))) GO TO 210
C         SET THE ERROR PRARMETER TO INDICATE THAT EXTRAPOLATION HAS OCCURRED.
  200     ERR=1
          NUM1=NUM-1
```

```
      CALL CHOOSE(XTAB(NUM1),YTAB(NUM1),DTAB(NUM1),XTAB(NUM),XTAB(NUM),YTAB(NUM),
     *          EPS,NCASE)
      CALL CASES(XTAB(NUM1),YTAB(NUM1),DTAB(NUM1),DTAB(NUM),XTAB(NUM),YTAB(NUM),
     *          E1,E2,V1,V2,W1,W2,Z1,Z2,Y1,Y2,NCASE)
  210 END1=END+1
      DO 220 I=END1,NUME
        YVAL(I)=SPLINE(XVAL(I),Z1,Z2,XTAB(NUM1),YTAB(NUM1),XTAB(NUM),YTAB(NUM),Y1,
       *          Y2,E2,W2,V2,NCASE)
  220 CONTINUE
C
  230 RETURN
      END
```

```
      SUBROUTINE SEARCH(XTAB, NUM, S, LCN, FND)
C
      REAL XTAB(NUM), S
      INTEGER FND, FIRST
C
C     SEARCH CONDUCTS A BINARY SEARCH FOR S. SEARCH IS CALLED ONLY IF S IS BETWEEN
C     XTAB(1) AND XTAB(NUM).
C
C     ON INPUT-
C
C     XTAB CONTAINS THE ABSCISSAS OF THE DATA POINTS OF INTERPOLATION.
C     NUM IS THE DIMENSION OF XTAB.
C     S IS THE VALUE WHOSE RELATIVE POSITION IN XTAB IS LOCATED BY SEARCH.
C
C     ON OUTPUT-
C
C     FND IS SET EQUAL TO 1 IF S IS FOUND IN XTAB AND IS SET EQUAL TO 0 OTHERWISE.
C     LCN IS THE INDEX OF THE LARGEST VALUE IN XTAB FOR WHICH XTAB(I).LT.S.
C
C     AND
C
C     SEARCH DOES NOT ALTER XTAB,NUM,S.
C
      FIRST = 1
      LAST = NUM
      FND = 0
C
      IF (XTAB(1).NE.S) GO TO 10
      LCN = 1
      FND = 1
      RETURN
C
   10 IF (XTAB(NUM).NE.S) GO TO 20
      LCN = NUM
      FND = 1
      RETURN
C
C     IF (LAST-FIRST) .EQ. 1, S IS NOT IN XTAB. SET POSITION EQUAL TO FIRST.
   20 IF ((LAST-FIRST).EQ.1) GO TO 30
C
      MIDDLE = (FIRST+LAST)/2
C
C     CHECK IF S .EQ. XTAB(MIDDLE). IF NOT, CONTINUE THE SEARCH IN THE APPROPRIATE
```

```
C           HALF OF THE VECTOR XTAB.
            IF (XTAB(MIDDLE)-S) 40, 50, 60
C
      30    LCN = FIRST
            RETURN
      40    FIRST = MIDDLE
            GO TO 20
      50    LCN = MIDDLE
            FND = 1
            RETURN
      60    LAST = MIDDLE
            GO TO 20
            END
```

```
      SUBROUTINE CHOOSE(P1, P2, D1, D2, Q1, Q2, EPS, NCASE)
C
      REAL P1,P2,D1,D2,Q1,Q2,DREF,DREF1,DREF2,SPQ,PROD,PROD1,PROD2,EPS
C
C     CHOOSE DETERMINES THE CASE NEEDED FOR THE COMPUTATION OF THE PARAMETERS OF
C     THE QUADRATIC SPLINE AND RETURNS THE VALUE IN THE VARIABLE NCASE.
C
C     ON INPUT-
C
C     (P1,P2) GIVES THE COORDINATES OF ONE OF THE POINTS OF INTERPOLATION.
C     D1 SPECIFIES THE DERIVATIVE CONDITION AT (P1,P2).
C     (Q1,Q2) GIVES THE COORDINATES OF ONE OF THE POINTS OF INTERPOLATION.
C     D2 SPECIFIES THE DERIVATIVE CONDITION AT (Q1,Q2).
C     EPS IS AN ERROR TOLERANCE USED TO DISTINGUISH CASES WHEN D1 OR D2 IS RELATIVELY
C     CLOSE TO THE SLOPE OR TWICE THE SLOPE OF THE LINE SEGMENT JOINING (P1,P2) AND
C     (Q1,Q2). IF EPS IS NOT EQUAL TO ZERO, THEN EPS SHOULD BE GREATER THAN OR EQUAL
C     TO MACHINE EPSILON.
C
C     ON OUTPUT-
C
C     NCASE CONTAINS THE VALUE WHICH CONTROLS HOW THE PARAMETERS OF THE QUADRATIC
C     SPLINE ARE EVALUATED.
C
C     AND
C
C     CHOOSE DOES NOT ALTER P1,P2,Q1,Q2,D1,D2,EPS.
C
C     CALCULATE THE SLOPE SPQ OF THE LINE JOINING (P1,P2),(Q1,Q2).
      SPQ = (Q2-P2)/(Q1-P1)
C
C     CHECK WHETHER OR NOT SPQ IS 0.
      IF (SPQ.NE.0.E0) GO TO 20
      IF ((D1*D2).GE.0.E0) GO TO 10
      NCASE = 1
      RETURN
   10 NCASE = 2
      RETURN
C
   20 PROD1 = SPQ*M1
      PROD2 = SPQ*M2
C
C     FIND THE ABSOLUTE VALUES OF THE SLOPES SPQ,D1 AND D2.
      DREF = ABS(SPQ)
```

```
            DREF1 = ABS(M1)
            DREF2 = ABS(M2)
C
C           IF THE RELATIVE DEVIATION OF D1 OR D2 FROM SPQ IS LESS THAN EPS, THEN
C           CHOOSE CASE 2 OR CASE 3.
            IF ((ABS(SPQ-D1).LE.EPS*DREF) .OR. (ABS(SPQ-D2).LE.EPS*DREF)) GO TO 30
C
C           COMPARE THE SIGNS OF THE SLOPES SPQ,D1 AND D2.
            IF ((PROD1.LT.0.E0).OR.(PROD2.LT.0.E0)) GO TO 80
            PROD = (DREF-DREF1)*(DREF-DREF2)
            IF (PROD.GE.0.E0) GO TO 40
C
C           L1, THE LINE THROUGH (P1,P2) WITH SLOPE D1, AND L2, THE LINE THROUGH (Q1,Q2)
C           WITH SLOPE D2, INTERSECT AT A POINT WHOSE ABSCISSA IS BETWEEN P1 AND Q1.
C           THE ABSCISSA BECOMES A KNOT OF THE SPLINE.
            NCASE = 1
            RETURN
C
   30       IF ((PROD1.LT.0.E0).OR.(PROD2.LT.0.E0)) GO TO 80
   40       IF (DREF1.GT.(2.E0*DREF)) GO TO 50
            IF (DREF2.GT.(2.E0*DREF)) GO TO 60
C
C           BOTH L1 AND L2 CROSS THE LINE THROUGH (P1+Q1/2,P2) AND (P1+Q1/2,Q2), WHICH IS
C           THE MIDLINE OF THE RECTANGLE FORMED BY (P1,P2), (Q1,P2), (Q1,Q2) AND (P1,Q2),
C           OR BOTH D1 AND D2 HAVE SIGNS DIFFERENT THAN THE SIGN OF SPQ, OR ONE OF D1 AND D2
C           HAS OPPOSITE SIGN FROM SPQ AND L1 AND L2 INTERSECT TO THE LEFT OF P1 OR TO THE
C           RIGHT OF Q1. THE POINT (P1+Q1)/2 IS A KNOT OF THE SPLINE.
            NCASE = 2
            RETURN
C
C           CHOOSE CASE 4 IF DREF2 IS GREATER THAN (2.-EPS)*DREF; OTHERWISE, CHOOSE CASE 3.
   50       IF (DREF2.GT.(2.E0-EPS)*DREF) GO TO 70
            NCASE = 3
            RETURN
C
C           IN CASES 3 AND 4, SIGN(M1)=SIGN(M2)=SIGN(SPQ).
C
C           EITHER L1 OR L2 CROSSES THE MIDLINE, BUT NOT BOTH.
C           CHOOSE CASE 4 IF MREF1 IS GREATER THAN (2.-EPS)*MREF; OTHERWISE, CHOOSE CASE 3.
   60       IF (DREF1.GT.(2.E0-EPS)*DREF) GO TO 70
            NCASE = 3
            RETURN
C
```

```
C               IF NEITHER L1 NOR L2 CROSSES THE MIDLINE, THE SPLINE REQUIRES TWO KNOTS
C               BETWEEN P1 AND Q1.
     70         NCASE = 4
                RETURN
C
C               THE SIGN OF AT LEAST ONE OF THE SLOPES D1,D2 DOES NOT AGREE WITH THE SIGN OF
C               THE SLOPE SPQ.
     80         IF ((PROD1.LT.0.E0).AND.(PROD2.LT.0.E0)) GO TO 130
C
                IF (PROD1.LT.0.E0) GO TO 90
                GO TO 110
C
     90         IF (DREF2.GT.((1.E0+EPS)*DREF)) GO TO 100
                NCASE = 2
                RETURN
C
    100         NCASE = 1
                RETURN
C
    110         IF (DREF1.GT.((1.E0+EPS)*DREF)) GO TO 120
                NCASE = 2
                RETURN
C
    120         NCASE = 1
                RETURN
C
    130         NCASE = 2
                RETURN
C
                END
```

```
      SUBROUTINE CASES(P1,P2,D1,D2,Q1,Q2,E1,E2,V1,V2,W1,W2,Z1,Z2,Y1,Y2,NCASE)
C
      REAL P1,P2,D1,D2,Q1,Q2,V1,V2,Z1,Z2,W1,W2,E1,E2,DBAR1,DBAR2,DBAR3,C1,M1,H1,J1,
     *     Y1,Y2,K1,ZTWO
C
C     CASES COMPUTES THE KNOTS AND OTHER PARAMETERS OF THE SPLINE ON THE
C     INTERVAL (P1,Q1).
C
C     ON INPUT-
C
C     (P1,P2) AND (Q1,Q2) ARE THE COORDINATES OF THE POINTS OF INTERPOLATION.
C     D1 IS THE SLOPE AT (P1,P2).
C     D2 IS THE SLOPE AT (Q1,Q2)
C     NCASE CONTROLS THE NUMBER AND LOCATION OF THE KNOTS.
C
C     ON OUTPUT-
C
C     (V1,V2),(W1,W2),(Z1,Z2), AND (E1,E2) ARE THE COORDINATES OF THE KNOTS AND
C     OTHER PARAMETERS OF THE SPLINE ON (P1,Q1). (E1,E2) AND (Y1,Y2) ARE USED
C     ONLY IF NCASE=4.
C
C     AND
C
C     CASES DOES NOT ALTER P1,P2,D1,D2,Q1,Q2.
C
      IF ((NCASE.EQ.3) .OR. (NCASE.EQ.4)) GO TO 20
      IF (NCASE.EQ.2) GO TO 10
C
C     CALCULATE THE PARAMETERS FOR CASE 1.
      Z1 = (P2-Q2+D2*Q1-D1*P1)/(D2-D1)
      ZTWO = P2 + D1*(Z1-P1)
      V1 = (P1+Z1)/2.E0
      V2 = (P2+ZTWO)/2.E0
      W1 = (Z1+Q1)/2.E0
      W2 = (ZTWO+Q2)/2.E0
      Z2 = V2 + ((W2-V2)/(W1-V1))*(Z1-V1)
      RETURN
C
C     CALCULATE THE PARAMETERS FOR CASE 2.
   10 Z1 = (P1+Q1)/2.E0
      V1 = (P1+Z1)/2.E0
      V2 = P2 + D1*(V1-P1)
      W1 = (Z1+Q1)/2.E0
```

```
      W2 = Q2 + D2*(W1-Q1)
      Z2 = (V2+W2)/2.E0
      RETURN
C
C     CALCULATE THE PARAMETERS USED IN BOTH CASES 3 AND 4.
   20 C1 = P1 + (Q2-P2)/D1
      D1 = Q1 + (P2-Q2)/D2
      H1 = 2.E0*C1 - P1
      J1 = 2.E0*M1 - Q1
      DBAR1 = (Q2-P2)/(H1-P1)
      DBAR2 = (P2-Q2)/(J1-Q1)
C
C     IF (NCASE.EQ.4) GO TO 50
C
C     CALCULATE THE PARAMETERS FOR CASE 3.
      K1 = (P2-Q2+Q1*DBAR2-P1*DBAR1)/(DBAR2-DBAR1)
      IF (ABS(D1).GT.ABS(D2)) GO TO 30
      Z1 = (K1+Q1)/2.E0
      GO TO 40
   30 Z1 = (K1+P1)/2.E0
   40 V1 = (P1+Z1)/2.E0
      V2 = P2 + D1*(V1-P1)
      W1 = (Q1+Z1)/2.E0
      W2 = Q2 + D2*(W1-Q1)
      Z2 = V2 + ((W2-V2)/(W1-V1))*(Z1-V1)
      RETURN
C
C     CALCULATE THE PARAMETERS FOR CASE 4.
   50 Y1 = (P1+C1)/2.E0
      V1 = (P1+Y1)/2.E0
      V2 = D1*(V1-P1) + P2
      Z1 = (M1+Q1)/2.E0
      W1 = (Q1+Z1)/2.E0
      W2 = D2*(W1-Q1) + Q2
      DBAR3 = (W2-V2)/(W1-V1)
      Y2 = DBAR3*(Y1-V1) + V2
      Z2 = DBAR3*(Z1-V1) + V2
      E1 = (Y1+Z1)/2.E0
      E2 = DBAR3*(E1-V1) + V2
      RETURN
C
      END
```

```
      SUBROUTINE SPLINE(XVALS,Z1,Z2,XTABS,YTABS,XTABS1,YTABS1,Y1,Y2,E2,W2,V2,NCASE)
C
      REAL XVALS,Z1,Z2,XTABS,YTABS,XTABS1,YTABS1,V2,W2,Y1,Y2,E2
C
C     SPLINE FINDS THE IMAGE OF A POINT IN XVAL.
C
C     ON INPUT–
C
C     XVALS CONTAINS THE VALUE AT WHICH THE SPLINE IS EVALUATED.
C     (XTABS,YTABS) ARE THE COORDINATES OF THE LEFT-HAND DATA POINT USED IN THE
C     EVALUATION OF XVALS.
C     (XTABS1,YTABS1) ARE THE COORDINATES OF THE RIGHT-HAND DATA POINT USED IN THE
C     EVALUATION OF XVALS.
C     Z1,Z2,Y1,Y2,E2,W2,V2 ARE THE PARAMETERS OF THE SPLINE.
C     NCASE CONTROLS THE EVALUATION OF THE SPLINE BY INDICATING WHETHER ONE OR TWO
C     KNOTS WERE PLACED IN THE INTERVAL (XTABS,XTABS1).
C
C     ON OUTPUT–
C
C     SPLINE IS THE IMAGE OF XVALS.
C
C     AND
C
C     SPLINE DOES NOT ALTER ANY OF THE INPUT PARAMETERS.
C
C     IF NCASE .EQ. 4, MORE THAN ONE KNOT WAS PLACED IN THE INTERVAL.
      IF (NCASE.EQ.4) GO TO 40
C
C     CASES 1,2, OR 3.
C
C     DETERMINE THE LOCATION OF XVALS RELATIVE TO THE KNOT.
      IF (Z1-XVALS) 10, 20, 30
C
   10 SPLINE = (Z2*(XTABS1-XVALS)**2+W2*2.E0*(XVALS-Z1)*(XTABS1-XVALS)
     *        +YTABS1*(XVALS-Z1)**2)/(XTABS1-Z1)**2
      RETURN
C
   20 SPLINE = Z2
      RETURN
C
   30 SPLINE = (YTABS*(Z1-XVALS)**2+V2*2.E0*(XVALS-XTABS)*(Z1-XVALS)+Z2*
     *        (XVALS-XTABS)**2)/(Z1-XTABS)**2
```

```
            RETURN
C
C           CASE 4.
C
C           DETERMINE THE LOCATION OF XVALS RELATIVE TO THE FIRST KNOT.
   40       IF (Y1-XVALS) 70, 60, 50
C
   50       SPLINE = (YTABS*(Y1-XVALS)**2+V2*2.E0*(XVALS-XTABS)*(Y1-XVALS)+Y2*
            *          (XVALS-XTABS)**2)/(Y1-XTABS)**2
            RETURN
C
   60       SPLINE = Y2
            RETURN
C
C           DETERMINE THE LOCATION OF XVALS RELATIVE TO THE SECOND KNOT.
   70       IF (Z1-XVALS) 100, 90, 80
C
   80       SPLINE = (Y2*(Z1-XVALS)**2+E2*2.E0*(XVALS-Y1)*(Z1-XVALS)+Z2*
            *          (XVALS-Y1)**2)/(Z1-Y1)**2
            RETURN
C
   90       SPLINE = Z2
            RETURN
C
  100       SPLINE = (Z2*(XTABS1-XVALS)**2+W2*2.E0*(XVALS-Z1)*(XTABS1-XVALS)
            *          +YTABS1*(XVALS-Z1)**2)/(XTABS1-Z1)**2
            RETURN
C
            END
```

A.2 Algorithme de Schumaker

```
/*****************************************************************************************
Ce programme a pour but d'implanter l'algorithme de Larry L. Schumaker, et de l'améliorer
*****************************************************************************************/

/* INCLUSIONS */

#include <fstream.h>
#include <math.h>
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* DÉCLARATION DES CONSTANTES ET DES FONCTIONS */

#define ABS(a) ((a) >= 0.0? (a) : -(a))
double F (double A[ ], double B[ ], double C[ ], double x[ ], int k, double var);

/* DÉCLARATION DES VARIABLES */

ifstream lecture;
int i ,j ,n, k1, k2, k3, k;
char fichier_entree[40];
double *V, *x, *y, *d, *L, *LT, *del, *t, *dbar, *A, *B, *C, *xi, *sbar, *eta, a, b, sd, sg, h, var;

/* DÉBUT DU PROGRAMME PRINCIPAL */

int main(){

        cout<<"Entrez le nom du fichier d'entree: ";
        cin>>fichier_entree;

/* LECTURE DU FICHIER D'ENTRÉE */

        lecture.open(fichier_entree);

        if(lecture.fail())
            cout<<"Impossible d'ouvrir le fichier "<<fichier_entree<<endl;
        else
            {
                lecture>>n;
```

```
                cout<<"n = "<<n<<endl;
                V = new double[2 * n];              //les données
                i = 0;
                while( !lecture.eof() )
                    {
                        lecture>>V[i];
                        i++;
                    }
            }

        lecture.close();

/* INITIALISATION */

    x   =   new double[2 * n - 1];      //les abscisses
    y   =   new double[2 * n - 1];      //les ordonnées
    d   =   new double[2 * n - 1];      //les pentes
    L   =   new double[n];              //les longueurs des cordes de même pentes
   LT   =   new double[n];              //variable temporaire pour le calcul de L
  del   =   new double[n];              //les pentes des cordes
    t   =   new double[2 * n - 1];      //variable temporaire pour le calcul de x
 dbar   =   new double[2 * n - 1];      //variable temporaire pour le calcul de d
    A   =   new double[2 * n - 1];      //les coefficients de la spline
    B   =   new double[2 * n - 1];
    C   =   new double[2 * n - 1];
   xi   =   new double[n - 1];          //les noeuds additionnels
 sbar   =   new double[n - 1];          //la dérivée au noeud
  eta   =   new double[n - 1];          //l'image par la spline au noeud

/* PREPROCESSING */

        for (i=0; i<n; i++)
            {
                x[i]=V[2 * i];
                y[i]=V[2 * i + 1];
            }
        delete V;

        for (i=0; i<n-1; i++)
            {
                L[i] = sqrt((x[i+1]-x[i])*(x[i+1]-x[i])+(y[i+1]-y[i])*(y[i+1]-y[i]));
                del[i] = (y[i+1]-y[i]) / (x[i+1]-x[i]);
            }
```

```
        for (i=0; i<n-1; i++)
            {
                j = i;
                sd = sg = 0.0;
                while( (del[j] == del[j+1])&&(j<n-1) )
                    {
                        sd = sd+L[j+1];
                        j++;
                    }
                j = i;
                while( (del[j] == del[j-1])&&(j>0) )
                    {
                        sg = sg+L[j-1];
                        j--;
                    }
                LT[i] = L[i]+sd+sg;
            }

        for (i=0; i<n-1; i++)
            L[i]=LT[i];

        delete LT;

/* CALCUL DES PENTES */

        for (i=1; i<n-1; i++)
            d[i]=(L[i-1]*del[i-1]+L[i]*del[i])/(L[i-1]+L[i]);

        delete L;

/* LEFT END SLOPE */

        d[0] = (3*del[0]-d[1])/2;

/* RIGHT END SLOPE */

        d[n-1] = (3*del[n-2]-d[n-2])/2;

/* COMPUTE KNOTS AND COEFFICIENTS */

        j=k1=k2=k3=0;
        for (i=0; i<n-1; i++)
            {
```

```
            if( d[i]+d[i+1] == 2*del[i] )
                {
                    t[j] = x[i]; dbar[j]=d[i]; k1++;
                    A[j] = y[i];
                    B[j] = d[i];
                    C[j] = (d[i+1]-d[i])/(2*(x[i+1]-x[i]));
                    j = j+1;
                }
            else
                {
                    a = d[i]-del[i];
                    b = d[i+1]-del[i];
                    if ( a*b >= 0 )
                        xi[i] = (x[i+1]+x[i])/2;
                    else
                        {
                            if ( ABS(a) < ABS(b) )
                                xi[i]=x[i+1]+a*(x[i+1]-x[i])/(d[i+1]-d[i]);
                            else
                                xi[i]=x[i]+b*(x[i+1]-x[i])/(d[i+1]-d[i]);
                        }
                    sbar[i]=(2*del[i]-d[i+1])+(d[i+1]-d[i])*(xi[i]-x[i])/(x[i+1]-x[i]);
                    eta[i]=(sbar[i]-d[i])/(xi[i]-x[i]);
                    t[j]=x[i]; dbar[j]=d[i]; k2++;
                    A[j]=y[i];
                    B[j]=d[i];
                    C[j]=eta[i]/2;
                    j=j+1;
                    t[j]=xi[i]; dbar[j]=sbar[i]; k3++;
                    A[j]=y[i]+d[i]*(xi[i]-x[i])+eta[i]*(xi[i]-x[i])*(xi[i]-x[i])/2;
                    B[j]=sbar[i];
                    C[j]=(d[i+1]-sbar[i])/(2*(x[i+1]-xi[i]));
                    j = j+1;
                }
        }

    delete del;
    delete xi;
    delete sbar;
    delete eta;

/* PRÉSENTATION DES NOEUDS */
```

```
        k=k1+k2+k3;
        x[k] = x[n-1];
        d[k] = d[n-1];

        for (j=0; j<k; j++)
            {
                x[j] = t[j];
                d[j] = dbar[j];
            }

        delete t;
        delete dbar;

        y[k] = y[n-1];

        for (j=0; j<k; j++)
            y[j] = F(A,B,C,x,k,x[j]);

}          /* fin MAIN */

/* DÉFINITION DE LA FONCTION "F" */

double F (double A[ ], double B[ ], double C[ ], double x[ ], int k, double var)
{
        int i;
        double rep;

        for (i=0; i<k; i++)
            {
                if ( (var<x[i+1])&&(var>=x[i]) )
                    rep=A[i]+(var-x[i])*(B[i]+C[i]*(var-x[i]));
            }

        return rep;
}
```

BIBLIOGRAPHIE

[1] H. Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM* **17**, pp. 589-602, 1970.

[2] R. K. Beatson et H. Wolkowicz. Postprocessing piecewise cubics for monotonicity. *SIAM J. Numer. Anal.* **26**, pp. 480-502, 1989.

[3] J. Butland. A method of interpolating reasonably-shaped curves through any data. *Proc. Computer Graphics* **80**, Online Publication Ltd, Middlesex, U.K., pp. 409-422, 1980.

[4] P. Costantini. An algorithm for computing shape-preserving splines of arbitrary degree. *J. Comput. Appl. Math.* **22**, pp. 89-136, 1988.

[5] C. de Boor. A Practical Guide to Splines. Springer-Verlag, New York, 1978.

[6] R. A. DeVore et Z. Yan. Error analysis for piecewise quadratic curve fitting algorithms. *CAGD* **3**, pp. 205-215, 1986.

[7] F. Dubeau et J. Savoie. Periodic Quadratic Spline Interpolation. *J. Approx. Theory* **39**, pp. 77-88, 1983.

[8] W. H. Frey. A Useful Variant of McLaughlin's Interpolant. Technical Report No. GMR-5004, General Motors Research Laboratory, Warren, Michigan, 1985.

[9] F. N. Fritsch et R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal.* **17**, pp. 238-246, 1980.

[10] R. Iqbal. A one-pass algorithm for shape-preserving quadratic spline interpolation. *J. Sci. Comput.* **7**, pp. 359-376, 1992.

[11] A. Lahtinen. On the construction of monotony preserving taper curves. *Acta For. Fennica* **203**, pp. 1-34, 1988.

[12] A. Lahtinen. Shape-preserving interpolation by quadratic splines. *J. Comput. Appl. Math.* **29**, pp. 15-24, 1990.

[13] A. Lahtinen. On the choice of parameters in shape-preserving quadratic spline interpolation. *J. Comput. Appl. Math.* **39**, pp. 109-113, 1992.

[14] G. G. Lorentz. Bernstein Polynomials. University of Toronto Press, Toronto, 1953.

[15] D. F. McAllister et J. A. Roulier. An algorithm for computing a shape-preserving osculatory quadratic spline. *ACM Trans. Math. Software* **7**, pp. 331-347, 1981.

[16] D. F. McAllister et J. A. Roulier. Algorithm 574, shape-preserving osculatory quadratic splines [E1, E2]. *ACM Trans. Math. Software* **7**, pp. 384-386, 1981.

[17] L. L. Schumaker. On shape-preserving quadratic spline interpolation. *SIAM J. Numer. Anal.* **20**, pp. 854-864, 1983.

[18] L. L. Schumaker. Spline Functions: Basic Theory. Wiley-Interscience, New York, 1981.

[19] Z. Yan. Piecewise cubic fitting algorithm. *Math. Comput.* **49**, pp. 203-213, 1987.